

Manual de usuario del Generador Estocástico de Series Sintéticas GAMWGEN

Alessio Bocco (boccoalessio@gmail.com)
Daniel Bonhaure (danielbonhaure@gmail.com)
Guillermo Podestá (gpodesta@rsmas.miami.edu)

09 de October de 2020

Abstract

Este paquete contiene la implementación de un generador estocástico diario y multi-sitio de series meteorológicas sintéticas. La generación de series sintéticas es un insumo básico para el análisis probabilista de las sequías y sus impactos en el sector agrícola. El generador desarrollado es muy flexible y capaz de generar secuencias de valores diarios de precipitación y temperaturas máxima y mínima. A partir de estas últimas se pueden derivar otras variables como la radiación solar y la evapotranspiración.

El generador tiene dos variantes, una permite generar series para localidades puntuales, generador local, y otra que permite la generación en grillas regulares o en localidades arbitrarias, generador espacial. Esto la de la posibilidad al usuario de generar productos a medida para distintas aplicaciones. Además, el generador es capaz de utilizar variables auxiliares que producen simulaciones condicionadas para su uso en conjunto con modelos de cambio climático o de pronósticos estacionales para evaluar impactos en el largo o corto plazo, respectivamente.

Esta flexibilidad se sustenta en el uso modelos generalizados aditivos (GAM) que permiten modelar con mucha precisión el comportamiento de las variables meteorológicas y capturar las propiedades estadísticas de los datos observados. La modelación de las variables meteorológicas se divide en dos: por un lado, la ocurrencia y monto de precipitación y por otro, las temperaturas máxima y mínima. La ocurrencia de precipitación se modela a través de un modelo probit mientras que al monto se lo hace a través de una distribución aleatoria Gamma. Para la temperatura se utiliza un modelo autorregresivo condicionado por la ocurrencia de lluvia. A su vez, estos modelos pueden ser espacialmente correlacionados con campos aleatorios Gaussianos que contemplan la variabilidad espacial y temporal regional.

Además del generador, se incluyen diagnósticos estadísticos y gráficos para verificar la bondad de ajuste estadística de los GAM y validar que las series sintéticas sean consistentes con los registros históricos. Los diagnósticos son exhaustivos e incluyen todas las propiedades de las series que podrían afectar el desempeño de las mismas durante el análisis probabilista.

Contents

1	Introducción	2
2	Fundamento	4
3	Tipos de series sintéticas	5
4	Metodología	8
4.1	Introducción a los Modelos Aditivos Generalizados	8
4.1.1	Interpretabilidad vs Complejidad	8
4.2	Modelación	18
4.2.1	Clima local	19
4.2.2	Tiempo local	22
4.3	Tipos de modelos	26
5	Aplicación	28
5.1	Instalar paquetes necesarios	28
5.2	Creación de directorios	28
5.3	Generación de series sintéticas sin autocorrelación espacial	28
5.3.1	Creación de archivos de entrada	29
5.3.2	Series sintéticas estacionarias	32
5.3.3	Series sintéticas pseudohistóricas	49
5.3.4	Series sintéticas correlacionadas espacialmente	65
5.4	Generación de series sintéticas para una grilla regular	81
References		105

1 Introducción

² El análisis de series climáticas es siempre un desafío y más aún en regiones donde la disponibilidad de las mismas es escasa. Este déficit es un problema especialmente importante para la caracterización del riesgo de desastres naturales. Este tipo de aplicaciones demandan

5 largas series climáticas para la realización de análisis cuantitativos del riesgo de desastres
6 que son claves para la estimación de su recurrencia y de las pérdidas asociadas. Las series
7 climáticas deben capturar la variabilidad natural del clima de la región de interés.

8 Al tratarse de un insumo necesario para el análisis probabilista de sequías el Sistema de
9 Información sobre Sequías para el sur de Sudamérica (SISSA) desarrolló un generador de
10 datos climáticos sintéticos. Los generadores estocásticos de clima producen series diarias de
11 variables climáticas con propiedades estadísticas consistentes a las de los datos observados.
12 Las principales variables de importancia son temperaturas máxima y mínima y precipitación
13 diaria. En muchas regiones estos datos se encuentran incompletos o son directamente inex-
14 istentes. Los registros suelen tener una duración insuficiente o sólo estar disponibles agre-
15 gados mensualmente. La cobertura espacial es otro de los problemas más comunes ya que,
16 en general, las redes de observación meteorológica son poco densas aún en zonas donde la
17 información meteorológica es de vital importancia.

18 La mayoría de los enfoques tradicionales para la generación de series estocásticas están lim-
19 itados por su capacidad de generar datos solamente en localidades para las que se cuenta
20 con observaciones (por ejemplo, donde existen estaciones meteorológicas). Otra desventaja
21 de algunas de estas herramientas (sobre todo los generadores no paramétricos basados en
22 remuestreo de observaciones) es que solamente pueden producir valores dentro del rango
23 observado en el registro histórico. En este proyecto, el generador desarrollado se basó en el
24 modelo diseñado por Verdin *et al.* (2016). Este generador estocástico diario multivariado
25 produce series sintéticas de precipitación y temperaturas máxima y mínima. El generador de
26 Verdin *et al.* (2016) utiliza cuatro modelos estadísticos para modelar la ocurrencia y montos
27 de precipitación y temperatura máxima y mínima basados en Modelos Lineales General-
28 izados (GLMs). Estos modelos son paramétricos y utilizan regresiones lineales entre las
29 variables para modelarlas. El proceso de ocurrencia de precipitación se modela como una re-
30 gresión probit mientras que los montos se ajustan a una distribución aleatoria Gamma. Las
31 temperaturas máximas y mínimas son consideradas variables autorregresivas condicionadas

32 por la precipitación. Sin embargo, una de las limitaciones fundamentales del generador de
33 Verdin *et al.* (2016) es que las temperaturas máxima y mínima se generan en forma indepen-
34 diente para cada día, por lo cual la amplitud térmica diaria simulada a veces no es realista.
35 Los diagnósticos realizados en base al generador de Verdin *et al.* (2016) demostraron que
36 algunas propiedades de las series sintéticas producidas no reflejaban fielmente características
37 importantes para el análisis de las sequías, por ejemplo, la persistencia de secuencias de días
38 secos y lluviosos. Por este motivo, en este trabajo solo se mantuvo la estructura general del
39 modelo de Verdin *et al.* (2016) y se modificaron todos los algoritmos (modelos estadísticos)
40 para obtener series sintéticas más consistentes con los registros históricos.

41 2 Fundamento

42 A partir de los datos climáticos históricos comienza el proceso de ajuste de un modelo
43 estadístico que permita representar el comportamiento de cada una de las variables para
44 cada localidad. El generador está dividido en cuatro modelos aditivos generalizados: dos para
45 modelar la precipitación y dos para las temperaturas máxima y mínima, respectivamente.
46 El concepto principal detrás de la generación de series sintéticas es que cada valor de una
47 variable puede ser considerado como la suma de una componente climática (clima local) y
48 otra componente meteorológica aleatoria o tiempo local (Kleiber *et al.*, 2013), es decir

$$X_{i,s} = \text{clima local} + \text{tiempo local}$$

49 donde $X_{i,s}$ corresponde al valor de la variable X en el día i en la localidad s ; clima local
50 corresponde a un valor medio de la variable para el día i en la localidad s y tiempo local
51 corresponde a un estado particular de la atmósfera en el día i en la localidad s . El componente
52 climático es especificado a través del ajuste de cada uno de los cuatro modelos aditivos
53 del generador. El componente meteorológico corresponde a los residuos de los modelos (la

54 diferencia entre los valores históricos y el componente climático estimado), es decir, a la
55 variabilidad no explicada por los mismos. El proceso de ajuste culmina con los parámetros
56 ajustados para los cuatro modelos mencionados. Posteriormente, se generan datos para los
57 años a simular que corresponden a los valores medios de la distribución para cada día del año
58 (clima local). Luego, se simulan una serie de valores aleatorios, a partir de los residuos de
59 cada modelo, que corresponden a la variabilidad propia de cada realización (tiempo local).
60 La Figura 1 muestra un ejemplo de ambos componentes para la temperatura máxima diaria
61 del año 1961 en la localidad de Junín (Argentina).

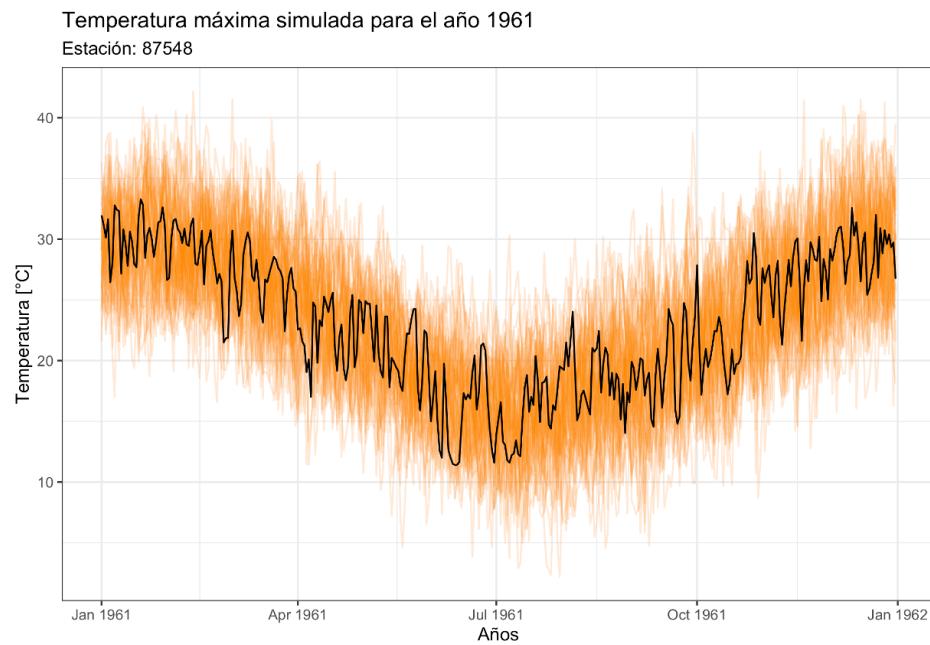


Figure 1: Componentes clima local y tiempo local

62 3 Tipos de series sintéticas

63 El generador GAMWGEN produce distintos tipos de series sintéticas. Desde el punto de
64 vista temporal, las series pueden ser **no condicionadas** (Figura 2), es decir, puramente
65 estacionarias; **pseudohistóricas** (Figura 3), que copian la variabilidad de baja frecuencia
66 y los cambios en la serie climática observada y **condicionadas** (Figura 4) que son series

67 forzadas a seguir la trayectoria de una salida de un modelo de cambio climático o una
68 trayectoria arbitraria definida por el usuario. Las siguientes Figuras muestran distintos
69 ejemplos de lo mencionado.

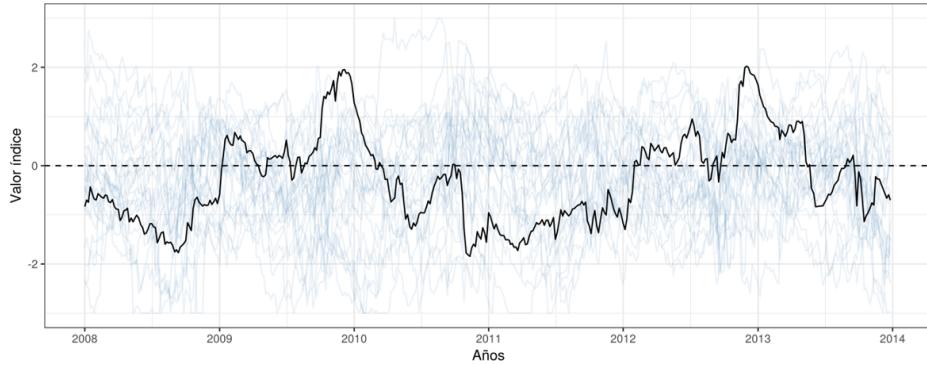


Figure 2: Tipos de series sintéticas: Series estacionarias

70 La línea negra corresponde a la serie temporal observada mientras que las distintas líneas
71 celestes corresponden a realizaciones del generador. Al tratarse de series puramente estocás-
72 ticas, son independientes entre sí.



Figure 3: Tipos de series sintéticas: Series pseudohistóricas

73 La línea negra corresponde a la serie temporal observada mientras que las distintas líneas
74 celestes corresponden a realizaciones del generador. En este caso, el generador fue forzado
75 a seguir la trayectoria en los datos observados, por lo tanto, las series generadas siguen las
76 variaciones de los datos observados.

77 En esta configuración el generador sigue una trayectoria arbitraria elegida por el usuario.

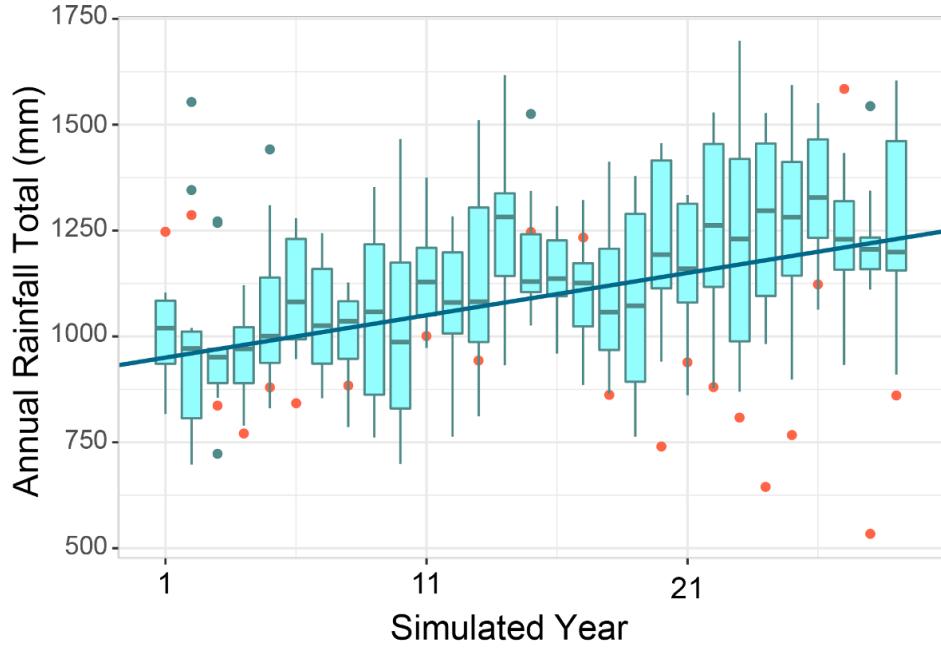


Figure 4: Tipos de series sintéticas: Series condicionadas

- 78 En este caso es una trayectoria lineal por lo que la precipitación aumenta un determinado
 79 porcentaje por año de manera lineal.
 80 Desde el punto de vista espacial se pueden generar series en estaciones meteorológicas, en
 81 puntos que no se correspondan con estaciones o en una grilla regular. La Figura 5 muestra
 82 un ejemplo de la generación en grillas sobre el Paraguay.

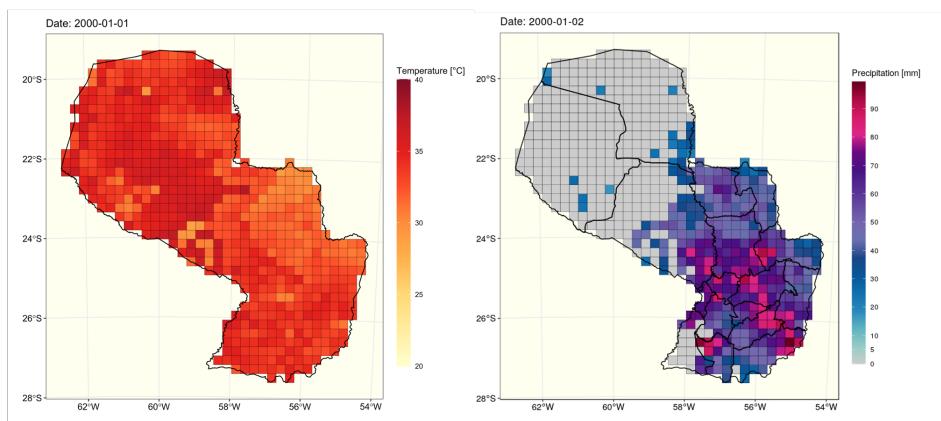


Figure 5: Datos grillados.

- 83 El panel de la izquierda muestra la temperatura máxima del día 1 de enero de 2000 para todo

⁸⁴ el territorio del Paraguay mientras que el mapa del panel derecho muestra la precipitación
⁸⁵ diaria para el mismo día.

⁸⁶ 4 Metodología

⁸⁷ Como su nombre indica, el generador está basado en Modelos Generalizados Aditivos (GAM,
⁸⁸ por sus siglas en inglés). Como se mencionó en le Introducción, este generador está basado
⁸⁹ en uno similar desarrollado por Verdin *et al.* (2016). Dicho generador estocástico utilizaba
⁹⁰ modelos lineales generalizados (GLM, por sus siglas en inglés). Los GLM son modelos muy
⁹¹ interesantes ya que son fatalmente interpretables aunque carecen de la flexibilidad necesaria
⁹² para capturar complejos patrones como las variaciones estacionales. Por ello, en esta versión
⁹³ se cambiaron los GLMs por GAMS. Estos nuevos modelos están siendo muy utilizados en
⁹⁴ diversas áreas porque heredan lo mejor de los GLM pero son mucho más flexibles. En la
⁹⁵ siguiente sección se describirán brevemente los GAMS.

⁹⁶ 4.1 Introducción a los Modelos Aditivos Generalizados

⁹⁷ 4.1.1 Interpretabilidad vs Complejidad

⁹⁸ La modelación estadística es una tarea muy compleja y que, en mnuchos casos, demanda un
⁹⁹ compromiso entre la interpretabilidad y complejidad de los modelos. La Figura 6 muestra
¹⁰⁰ tres alternativas con creciente nivel de complejidad.



Figure 6: Modelos estadísticos

101 En el extremo izquierdo, se encuentran los modelos lineales. Estos modelos son fáciles de
102 interpretar porque poseen una ecuación lineal explícita y, además, es muy sencillo hacer
103 inferencia a partir de ellos. Al tener una ecuación, cada variable tiene un coeficiente lineal
104 que permite entender la importancia de cada una de las variables involucradas en el modelo.
105 Sin embargo, en muchas aplicaciones es necesario modelar relaciones más complejas que sólo
106 lineales. En los casos en las que las relaciones son no lineales, un gran porcentaje de la
107 variabilidad se pierde y la inferencia realizada por dicho modelo sería muy pobre. En el otro
108 extremo del espectro hay toda una serie de modelos de tipo “caja negra” como las redes
109 neuronales, random forest, árboles de regresión, etc. Estos modelos son muy buenos para
110 modelar complejas relaciones pero son muy difíciles de interpretar y de entender qué está
111 sucediendo en el sistema. Son muy útiles para clasificar pero no sirven para entender como
112 una variable se relaciona con el resultado del modelo. Los GAMs proveen un interesante
113 punto intermedio ya que se pueden ajustar relaciones complejas, no lineales e interacciones.
114 Los modelos son explícitos y se pueden observar las relaciones entre variables y entender
115 porque se produce un resultado determinado.

116 **4.1.1.1 Relaciones no lineales** En general, las relaciones entre variables de la natu-
117 raleza no son lineales y adquieren patrones muy complejos. En la Figura 4 se muestra un
118 ejemplo de datos sintéticas para ejemplificar este concepto.

```
# Se simulan datos a partir de un GAM con distribución normal y  
# escala 0
```

```
# Se define una semilla para garantizar reproducibilidad  
set.seed(2)
```

```
# Se simulan 400 valores a partir del modelo  
dat <- mgcv::gamSim(1, n=400, dist = "normal",
```

```

    scale = 0, verbose = FALSE)

dat <- dat[,c("y", "x0", "x1", "x2", "x3")]

```

- 119 La Figura 7 muestra una diagrama de dispersión con valores simulados para ver su distribu-
 120 ción.

```

# Gráfico con los valores simulados

ggplot2::ggplot(dat, ggplot2::aes(y = y, x = x2)) +
  ggplot2::geom_point() +
  ggplot2::theme_minimal()

```

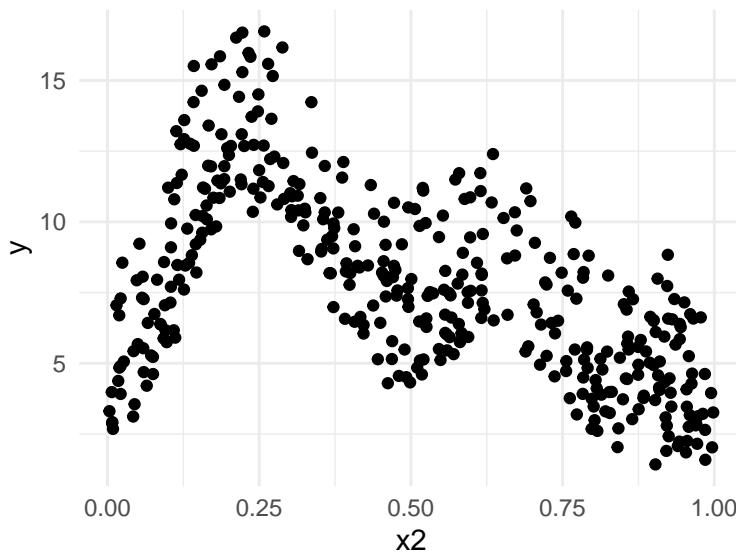


Figure 7: Relaciones no lineales.

- 121 Este diagrama de dispersión muestra dos variables que claramente se encuentran relacionadas
 122 pero no de manera lineal. Para continuar con el ejemplo de la sección anterior se ajustarán
 123 dos modelos a estos datos simulados, uno modlo lineal y uno generalizado aditivo.

 124 Al ajustar una regresión lineal simple a estos datos se obtiene una recta que atraviesa toda la
 125 nube de puntos. La Figura 8 muestra el resultado de la regresión. La línea azul corresponde
 126 al ajuste del modelo lineal y el área gris, al intervalo de confianza del 95%.

```

# Grafico con el ajuste lineal

ggplot2::ggplot(dat, ggplot2::aes(x2, y)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth(method = 'lm', se = TRUE, ggplot2::aes(colour = "Lineal")) +
  ggplot2::scale_colour_manual(name = "", values = c("Steelblue")) +
  ggplot2::theme_minimal() +
  ggplot2::theme(legend.position="none")

```

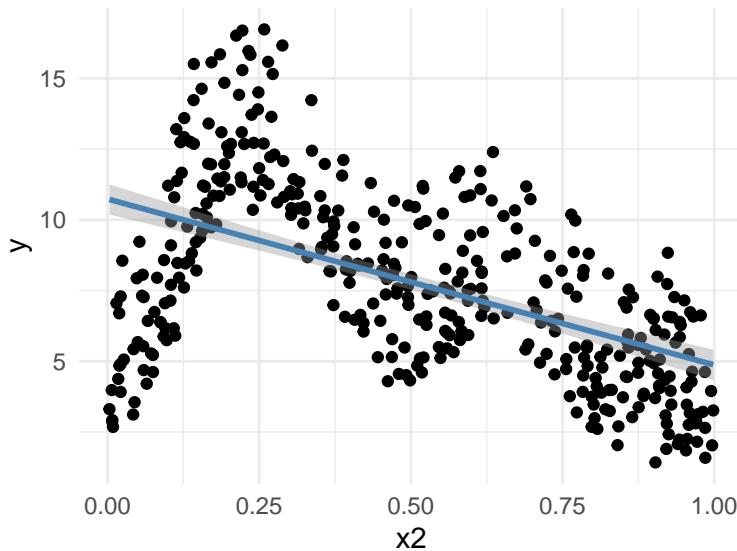


Figure 8: GAMs: Ajuste lineal de datos simulados.

¹²⁷ La regresión resultante representa la tendencia descendente en los datos simulados pero no
¹²⁸ es capaz de capturar las principales características de los datos como son las dos ondas con
¹²⁹ amplitud decreciente que se observa en la Figura 8.

¹³⁰ Modelo lineal

¹³¹ El modelo lineal se puede expresar mediante la siguiente ecuación:

$$y_i = \beta_0 + x_{1i}\beta_1 + \dots + \epsilon_i$$

¹³² donde, y_i corresponde a la variable respuesta y es una combinación lineal de las variables

133 regresoras; β_0 corresponde a la ordenada al origen, $x_{1i}\beta_1$ corresponde a la variable regreso β_1
 134 multiplicada por un coeficiente x_i que surge del ajuste del modelo y un término de error ϵ_i que
 135 usualmente tiene una distribución normal. Se pueden incluir más términos al modelo lineal,
 136 como términos cuadráticos o cúbicos, pero se corre el riesgo de sobreajustar severamente
 137 el modelo. En R el ajuste de los modelos lineales se realiza con la función `lm` cuyo primer
 138 argumento es la fórmula del modelo, en este caso: $\mathbf{y} \sim \mathbf{x2}$.

```

# Ajuste del modelo lineal
modelo_lineal <- lm(y ~ x2, data = dat)

# Evaluación de los resultados
summary(modelo_lineal)

139 ##
140 ## Call:
141 ## lm(formula = y ~ x2, data = dat)
142 ##
143 ## Residuals:
144 ##      Min       1Q   Median       3Q      Max
145 ## -8.0039 -1.8689 -0.0177  1.8640  7.5097
146 ##
147 ## Coefficients:
148 ##             Estimate Std. Error t value Pr(>|t|)
149 ## (Intercept) 10.7380     0.2729   39.34  <2e-16 ***
150 ## x2          -5.8684     0.4670  -12.57  <2e-16 ***
151 ## ---
152 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
153 ##
  
```

```

154 ## Residual standard error: 2.805 on 398 degrees of freedom
155 ## Multiple R-squared:  0.2841, Adjusted R-squared:  0.2823
156 ## F-statistic: 157.9 on 1 and 398 DF,  p-value: < 2.2e-16

```

157 Con la función `summary` se muestra una descripción del modelo ajustado así como una evaluación preliminar de la bondad del ajuste. En la sección *Coefficients* se muestra la significación de cada uno de los términos del modelo lineal, ordenada el origen (β_0) y pendiente (β_1), y se observa que ambos son altamente significativos. Sin embargo, al analizar el R^2 , sólo un 28% de la variabilidad es explicada por el modelo lineal. Lo anterior se comprueba al graficar los residuos del modelo. Los residuos son la diferencia entre los valores ajustados y observados.
 161 En la Figura 9 se muestra el diagnóstico de residuos vs valor ajustados, en el eje horizontal
 162 se ubican datos ajustados y en el eje vertical, los residuos del modelo.
 163

```

# Calculo de los residuos del modelo
residuals <- ggplot2::fortify(modelo_lineal)

# Gráfico de residuos
ggplot2::ggplot(residuals, aes(x = .fitted, y = .resid)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::geom_hline(yintercept = 0, linetype = 'dashed') +
  ggplot2::theme_bw() +
  ggplot2::xlab('Ajustados') + ggplot2::ylab('Residuos')

```

```

165 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

```

166 Si el modelo hubiera ajustado satisfactoriamente los datos, los residuos deberían distribuirse
 167 al azar y sin un patrón específico. Sin embargo, se observa en la Figura 9 que ésto no sucede

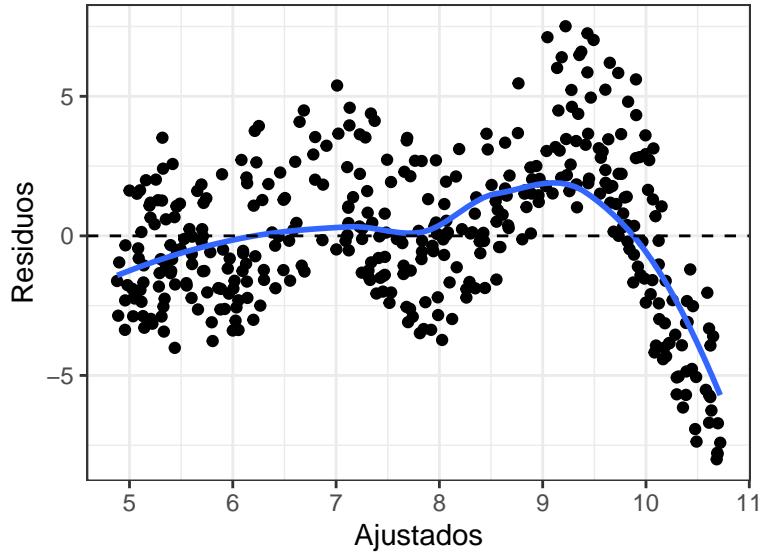


Figure 9: Residuos del modelo lineal.

¹⁶⁸ y que los puntos tiene un patrón muy marcado. Esto quiere decir que hay un gran porcentaje
¹⁶⁹ de la variabilidad que no es explicada por el modelo y por lo tanto, inferir sobre este modelo
¹⁷⁰ sería un grave error. Si en lugar de utilizar un modelo lineal, se ajustan los datos con un
¹⁷¹ `gam`, los resultados son muy diferentes. En la Figura 10 se muestra el ajuste del `gam` a los
¹⁷² datos observados.

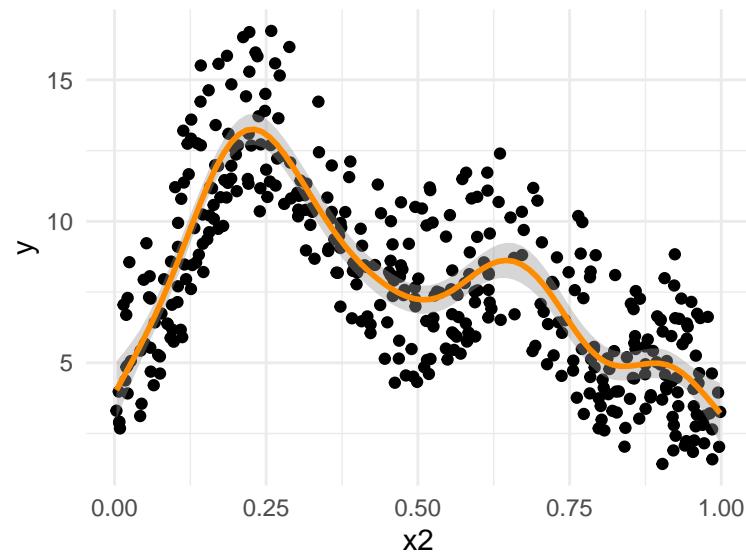


Figure 10: Ajuste no lineal.

173 La línea naranja corresponde al GAM ajustado, mientras que el área gris corresponde al
174 intervalo de confianza del 95%. A diferencia del modelo lineal, el GAM es mucho más
175 flexible y, no sólo sigue la tendencia descendente sino que captura las distintas ondas a pesar
176 de su distinta amplitud. Los GAMs ajustan el modelo a través de funciones suavizadas o
177 splines que pueden tomar casi cualquier forma. Al utilizar splines los GAMs pueden capturar
178 diversos tipos de relaciones no lineales y es por esto que son tan flexibles y adaptables a
179 distintos contextos. En el presente paquete el ajuste de los `gam` se realiza con el paquete
180 `mgcv` de Simon Wood (2015)

```
# Ajuste de un GAM con el paquete mgcv

modelo_gam <- mgcv::gam(y ~ s(x2), data = dat)

#  
summary(modelo_gam)

181 ##
182 ## Family: gaussian
183 ## Link function: identity
184 ##
185 ## Formula:
186 ## y ~ s(x2)
187 ##
188 ## Parametric coefficients:
189 ##              Estimate Std. Error t value Pr(>|t|)  
190 ## (Intercept) 7.796      0.093   83.82  <2e-16 ***
191 ## ---
192 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
193 ##
```

```

194 ## Approximate significance of smooth terms:
195 ##          edf Ref.df      F p-value
196 ## s(x2) 8.702 8.974 96.93 <2e-16 ***
197 ## ---
198 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
199 ##
200 ## R-sq.(adj) = 0.684 Deviance explained = 69.1%
201 ## GCV = 3.5458 Scale est. = 3.4598 n = 400

```

202 La función `summary` permite ver los resultados del ajuste del GAM. En este caso no hay un
203 sólo panel para los coeficientes como en el modelo lineal sino que en *Parametric coefficients* se
204 encuentra la ordenada al origen y bajo *Approximate significance of smooth terms* se muestra
205 la significancia del spline. Si bien el modelo no es perfecto, el R^2 aumentó del 20% a casi el
206 70%.

207 A diferencia de los modelos lineales o lineales generalizados donde la media de la variable
208 respuesta es una suma de los términos lineales:

$$y_i = \beta_0 + \sum_j \beta_j x_{ji} + \epsilon_i$$

209 Mientras que en un GAM, la media de la variable respuesta es una suma de *funciones*
210 *suavizadas* o *smooths*.

$$y_i = \beta_0 + \sum_j s_j(x_{ji}) + \epsilon_i$$

211 dónde, β_0 corresponde a la ordenada al origen; $\sum_j s_j(x_{ji})$ corresponde a la suma de las
212 *funciones suavizadas* y ϵ_i corresponde al término de error. Al realizar el gráfico de residuos
213 vs valores ajustados es posible verificar si el modelo ajustó tan bien como indica el valor de
214 R^2 . En la Figura 11 se muestra dicho gráfico.

```

# Obtener los residuos del GAM

residuals <- data.frame(fitted = modelo_gam$fitted.values,
                         resid = resid(modelo_gam)) %>%
  tibble::as_tibble()

# Gráfico de residuos vs valores ajustados

ggplot2::ggplot(residuals, aes(x = fitted, y = resid)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::geom_hline(yintercept = 0, linetype = 'dashed') +
  ggplot2::theme_bw() +
  ggplot2::xlab('Ajustados') + ggplot2::ylab('Residuos')

215 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

```

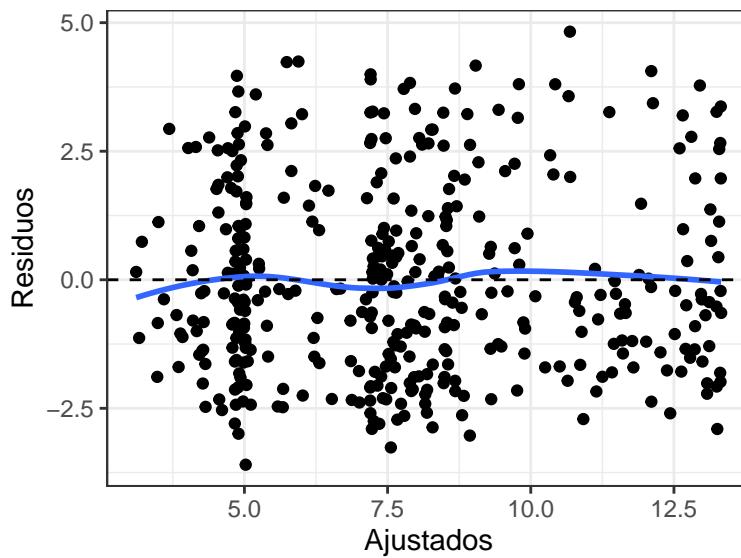


Figure 11: Residuos del GAM.

- 216 Al utilizar un GAM los residuos tienen una distribución casi aleatoria sin una tendencia
 217 clara como si fue el caso del modelo lineal. Este es sólo en sencillo ejemplo del potencial que

218 tienen los GAMs para modelar complejas relaciones. Si se desea profundizar más en el tema
219 se recomienda revisar el libro de Simon Wood (2017).

220 4.2 Modelación

221 Como se mencionó anteriormente, el generador estocástico tiene como base cuatro modelos
222 generalizados, dos para temperaturas máxima y mínima y dos para la precipitación que
223 modelan la ocurrencia y los montos diarios. La Figura 12 muestra una diagrama con el
224 proceso de ajuste de cada uno de ellos.

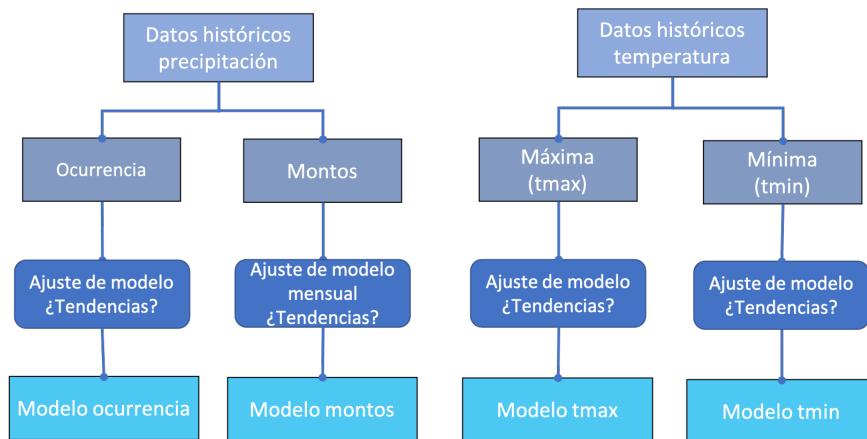


Figure 12: Modelos estadísticos

225 Partiendo de la base de datos se extraen los datos diarios de temperaturas máxima y míni-
226 ma y precipitación. Los valores de temperaturas son utilizados para ajustar un GAM que
227 modelará la componente climática. Estos modelos pueden incluir o no medias trimestrales
228 que servirán para condicionar el modelo a seguir una determinada trayectoria. Para el caso
229 de la precipitación el tratamiento es diferente. El fenómeno se divide en dos: ocurrencia de
230 precipitación y montos diarios. La ocurrencia se modela con GAM para toda la serie que
231 puede o no estar condicionado por totales trimestrales de lluvia. Los montos, en cambio, son
232 modelados a escala mensual. Es decir, se ajustan doce modelos, uno para cada mes del año.
233 Se utiliza esta modalidad para capturar mejor las características propias del ciclo estacional
234 de la precipitación de cada región.

²³⁵ A continuación se describirán cada uno de los modelos recién mencionados. La sección se
²³⁶ encuentra dividida por un lado se describirá la modelación del **clima local**

²³⁷ **4.2.1 Clima local**

²³⁸ **4.2.1.1 Ocurrencia de lluvia** El ajuste del modelo de ocurrencia comienza con la defini-
²³⁹ ción de un día lluvioso. Según la Organización Meteorológica Mundial (OMM), se considera
²⁴⁰ como día lluvioso a aquel día con una precipitación igual o mayor a 0.1 mm. Este valor,
²⁴¹ si bien es el sugerido por la OMM, puede ser modificado por el usuario si así lo dispone.
²⁴² Una vez definida la ocurrencia de lluvia se ajusta el modelo. La ocurrencia de lluvia es una
²⁴³ variable de tipo binaria, es decir, un día puede ser lluvioso (1) o seco (0), y es muy bien
²⁴⁴ representada a través de una regresión probit Kleiber *et al.* (2012). Este tipo de regresiones
²⁴⁵ se basan en procesos latentes Gaussianos, $W_{s,t}$, que se modelan con la siguiente relación:

$$O_{s,t} = \Pi_{\{W_{s,t} > 0\}}$$

²⁴⁶ Si el proceso $W_{s,t}$ es positivo, significa que lloverá en el día **t** y en la estación **s** y a ese día
²⁴⁷ se le asignará el valor 1. Si el proceso es negativo significa que no lloverá en el día **t** y en la
²⁴⁸ estación **s** y ese día se le asignará el valor 0. El uso de este tipo de procesos latentes está
²⁴⁹ justificado en que, en una región, la ocurrencia de lluvia en las distintas estaciones tenderá a
²⁵⁰ estar correlacionada. La función media del proceso latente Gaussiano es una regresión entre
²⁵¹ variables que se expresa de la siguiente manera:

$$O_{s,t} = (s, O_{s,t-1}, f(doy(t)), f(ST(t)), f(lon, lat))$$

²⁵² donde $O_{s,t}$ corresponde a la ocurrencia de lluvia en sitio y día determinado; s corresponde
²⁵³ al efecto del sitio **s** (ordenada al origen); $O_{s,t-1}$ corresponde a la ocurrencia del día previo
²⁵⁴ que es un término autorregresivo; $f(doy(t))$ corresponde a una función cíclica de los días

255 del año para considerar el efecto de la estacionalidad sobre la ocurrencia de lluvia; $f(ST(t))$
 256 corresponde a una función suavizada de los acumulados estacionales de precipitación (solo
 257 se utiliza si se desea condicionar el modelo) y $f(lon, lat)$ corresponde a las coordenadas del
 258 punto **s** (solo se utiliza en el modelo espacial). En la práctica, esta covariable se divide en
 259 cuatro, una para cada trimestre del año, asignándole el valor de 0 para los momentos fuera del
 260 respectivo trimestre. Es importante notar que para la ocurrencia de precipitación se usa un
 261 solo término autorregresivo. Este término es muy importante para la correcta modelización
 262 de las rachas secas y lluviosas mientras que el día del año incorpora la variabilidad intra-
 263 anual.

264 **4.2.1.2 Montos diarios de lluvia** El modelo de montos diarios de lluvia difiere del
 265 anterior en que no se usan todos los datos de la serie, sino que se extraen de la base de
 266 datos los montos de precipitación únicamente para los días lluviosos. La intensidad de
 267 la precipitación para una localidad s y tiempo t es modelada como una variable aleatoria
 268 Gamma cuyos parámetros de forma y escala varían en el tiempo y en el espacio (Kleiber *et al.*
 269 (2012)). La función Gamma ha sido ampliamente utilizada en la región para la modelación
 270 de acumulados de lluvia. El modelo de montos puede ser expresado de la siguiente manera:

$$I_{s,t} = (s, O_{s,t-1}, f(ST(t)), f(lon, lat))$$

271 donde, $I_{s,t}$ corresponde a los montos de precipitación en un sitio y día determinado; $O_{s,t-1}$
 272 corresponde a la ocurrencia del día previo para considerar la autocorrelación temporal;
 273 $f(ST(t))$ corresponde a una función suavizada de los acumulados estacionales de precip-
 274 itación (solo se utiliza si se desea condicionar el modelo) y $f(lon, lat)$ corresponde a las
 275 coordenadas del punto **s** (solo se utiliza en el modelo espacial). A diferencia del modelo
 276 anterior – que modela los años calendarios completos a través de la estacionalidad del día
 277 del año – la serie de montos diarios es dividida en función del mes del año para ajustar una

278 distribución a cada mes para obtener así parámetros mensuales más precisos. Gracias a esta
279 modificación se obtienen parámetros que varían en el tiempo y en el espacio lo que permite
280 capturar la variabilidad espacial de la precipitación. Al igual que en el modelo de ocurrencia,
281 la inclusión del total trimestral es necesaria si se desean simular series condicionadas.

282 **4.2.1.3 Temperatura** Para el caso de la temperatura se utiliza una metodología similar
283 a la utilizada para la precipitación, basada en Kleiber *et al.* (2013). A partir de los datos
284 observados de temperatura se ajustan dos modelos: uno de máxima y otro de mínima.
285 Ambos modelos utilizan las mismas variables para realizar el ajuste. El modelo puede ser
286 expresado de la siguiente manera:

$$X_{s,t} = (s, O_{s,t}, O_{s,t-1}, f(T_{x_{(s,t-1)}}, T_{n_{(s,t-1)}}), f(doy(t)), f(SX(t), SN(t)), f(lon, lat))$$

287 donde, $X_{s,t}$ corresponde a la temperatura máxima o mínima en un sitio y momento de-
288 terminado; s corresponde al efecto del sitio s (ordenada al origen); $O_{s,t}$ corresponde a la
289 ocurrencia de lluvia en sitio y día determinado; $O_{s,t-1}$ corresponde a la ocurrencia del día
290 previo; $f(T_{x_{(s,t-1)}}, T_{n_{(s,t-1)}})$ corresponde a una función suavizada de la interacción entre la
291 temperatura máxima y mínima del día previo; $f(doy(t))$ corresponde a una función cíclica
292 de los días del año para considerar el efecto de la estacionalidad sobre la temperatura; El
293 término $f(SX(t), SN(t))$ corresponde a la interacción entre las medias estacionales de tem-
294 peratura máxima y mínima y, como se explicó en la sección anterior, se incluyen para crear
295 modelos condicionados y $f(lon, lat)$ corresponde a las coordenadas del punto s (solo se uti-
296 liza en el modelo espacial). La ocurrencia de lluvia es muy importante ya que en general
297 los días lluviosos tienen temperaturas más bajas que los secos, sobre todo en verano, por lo
298 que debe ser incluido en el ajuste. La Figura 6 es un ejemplo de esta influencia en Junín,
299 en donde se muestra la amplitud térmica para cada mes del año en función del tipo de día,
300 seco o lluvioso.

301 **4.2.2 Tiempo local**

302 **4.2.2.1 Para una estación**

303 **4.2.2.1.1 Precipitación** El **tiempo local** de la ocurrencia de precipitación se modela a
304 través de los residuos de la regresión probit. Por definición estos residuos tienen una distribu-
305 ción $X \sim \mathcal{N}(0, 1)$. Por lo tanto, generar valores para el tiempo local es muy sencillo, solo
306 se necesita de una función gaussiana que genere números aleatorios. La Figura 13 muestra
307 un ejemplo de la generación de clima local y tiempo local para una estación meteorológica
308 de Argentina.

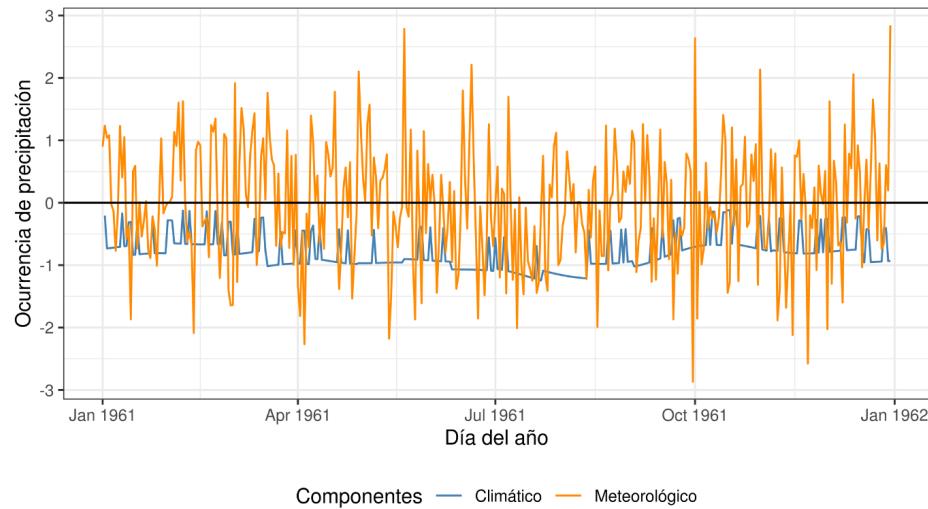


Figure 13: Tiempo local: precipitación

309 La Figura 13 muestra sólo un año para mejorar la visualización pero su interpretación es
310 válida para toda la longitud de la serie. La línea azul corresponde al clima local modelado
311 a través del GAM y la naranja al ruido aleatorio creado a partir de una distribución $X \sim$
312 $\mathcal{N}(0, 1)$. La sumatoria de ambos componentes determinarán si el día es lluvioso o no. Si la
313 suma es positiva, lloverá, caso contrario será un día seco. Se puede observar en la línea azul un
314 patrón estacional debido al régimen de precipitación tipo monzónico de esta región con picos
315 de más días lluviosos durante el verano mientras que en invierno disminuyen marcadamente.

³¹⁶ Esta misma serie temporal de números aleatorios serán la base para la generación de los
³¹⁷ montos de precipitación.

³¹⁸ **4.2.2.1.2 Temperatura** El tiempo local de las temperaturas máxima y mínima se mod-
³¹⁹ ela de una manera diferente. En este caso se toman los residuos de cada uno de los GAMs, es
³²⁰ decir, la diferencia entre el valor ajustado por el modelo y el valor observado de temperatura.
³²¹ Además, como la temperatura está fuertemente influenciada por el tipo de día, días lluviosos
³²² tienden a tener una menor amplitud térmica que los días secos, los residuos se separan en
³²³ función del tipo de día. Para capturar mejor el patrón estacional de la temperatura, los
³²⁴ residuos se agrupan por mes y se ajusta un modelo bivariado que contemple los residuos de
³²⁵ temperaturas máxima y mínima. El uso de un modelos bivariado es una alternativa muy
³²⁶ interesante para mantener la consistencia entre ambas variables, es decir, que la temperatura
³²⁷ máxima no supere a la mínima. La siguiente Figura 14 es un ejemplo de las series de tiempo
³²⁸ local para una localidad de Argentina.

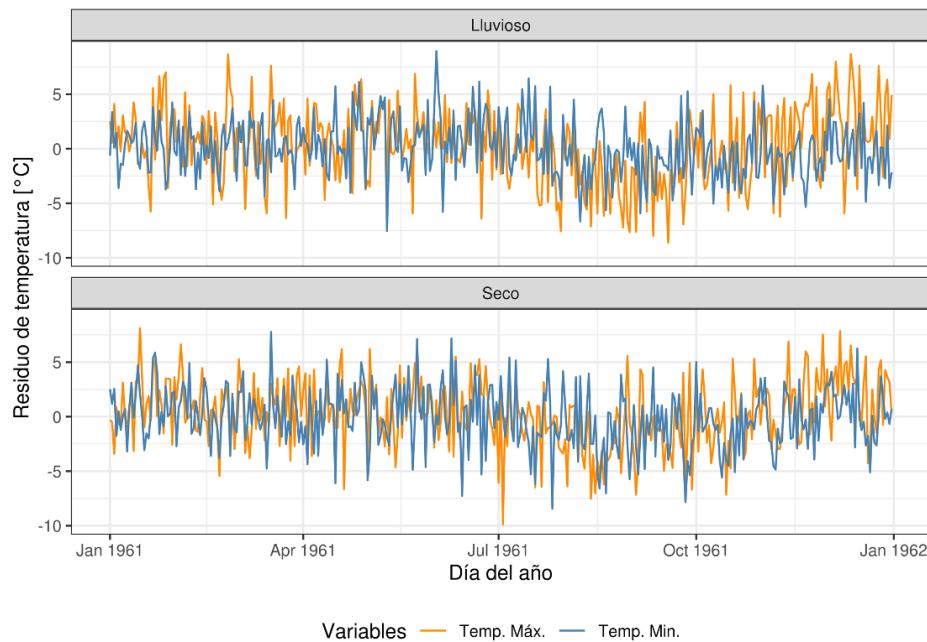


Figure 14: Tiempo local: temperatura

³²⁹ La Figura 14 esta dividida en dos paneles, el superior muestra el tiempo local para los días

330 lluviosos y el inferior para los días secos. La línea naranja corresponde a los valores de tiempo
331 local para temperatura máxima y los azules para temperatura mínima. Si bien ambas series
332 difieren en magnitud y variabilidad, las dos tienden a variar conjuntamente.

333 **4.2.2.2 Para una grilla** Para generar datos sobre una grilla regular o en puntos donde
334 no hay datos para ajustar los modelos se debe utilizar el modelo espacial. La generación
335 del tiempo local en el espacio es conceptualmente idéntico a la generación sobre estaciones
336 meteorológicas sólo que utiliza campos gaussianos para incluir la dimensión espacial. Los
337 campos gaussianos se generan con el paquete **RandomFields** (Schlather, Martin (2015)) y
338 capturan la variabilidad espacial de cada una de las variables a partir de su variograma.
339 ##### Precipitación

340 Para la precipitación se utiliza un modelo exponencial que utiliza como parámetros el vari-
341 ograma ajustado a partir de los datos observados usando máxima verosimilitud. Este modelo
342 permite simular campos con una muy buena consistencia espacial. La ocurrencia de precip-
343 itación no ocurre de manera aislada en una región sino que, en general, un evento lluvioso
344 abarca una importante superficie. La siguiente Figura 15 es una ejemplo de los campos
345 aleatorios generados sobre una grilla regular para una región de Argentina.

346 La interpretación es análoga a la mostrada para una estación puntual. Los valores para cada
347 píxel se suman a la componente climática y así se determina si el día será lluvioso o no.
348 Este tipo de campos se generan para todos los días de la simulación y cada campo diario es
349 independiente del anterior.

350 **4.2.2.2.1 Temperatura** Al igual que para una estación, los campos gaussianos que se
351 generan son bivariados. Se utiliza el modelo Bivariado de Whittle Matern incluido en el
352 paquete **RandomFields** (Schlather, Martin (2015)). La siguiente Figura 16 es un ejemplo
353 para un día de una realización para grilla regular sobre Argentina.

354 Ambos campos se generan simultáneamente para temperaturas máxima y mínima y discrim-

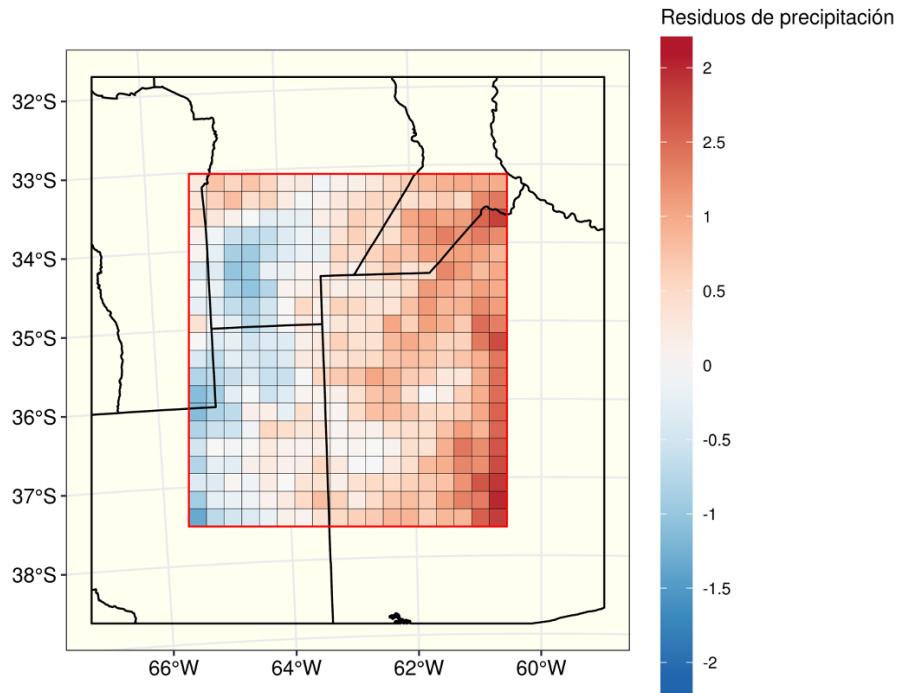


Figure 15: Tiempo local: precipitación sobre una grilla

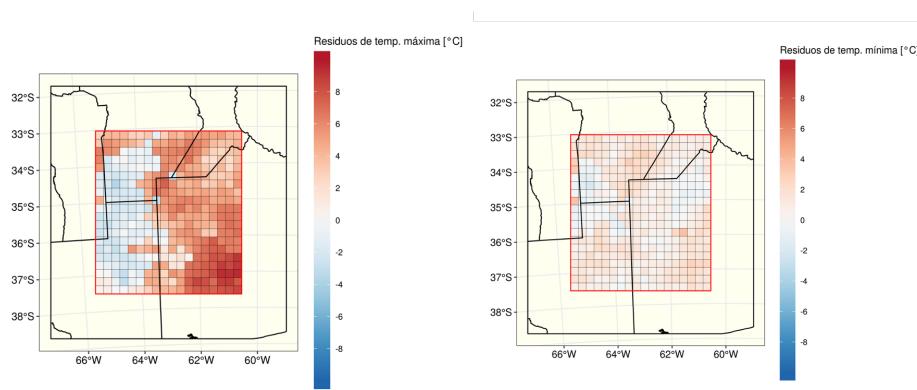


Figure 16: Tiempo local de temperatura sobre una grilla

355 inando entre días secos y lluviosos. Esto quiere decir que en el proceso de creación de los
356 campos se generan cuatro capas diferentes que luego se combinan en función del tipo de día
357 de cada píxel resultando en dos campos integradores. Los valores para cada píxel se sumen
358 a la componente climática y así se obtiene el valor final para cada día de la simulación.

359 4.3 Tipos de modelos

360 Basado en lo descrito en secciones anteriores, el generador cuenta con dos variantes: una
361 es capaz de simular en una grilla o en localidades arbitrarias, y la otra variante solo puede
362 generar series para sitios donde existen datos históricos y que fueron utilizados en el proceso
363 de ajuste. La Figura 17 muestra un diagrama del flujo de las dos variantes del generador,
364 que de aquí en más se denominarán modelo espacial y local, respectivamente.

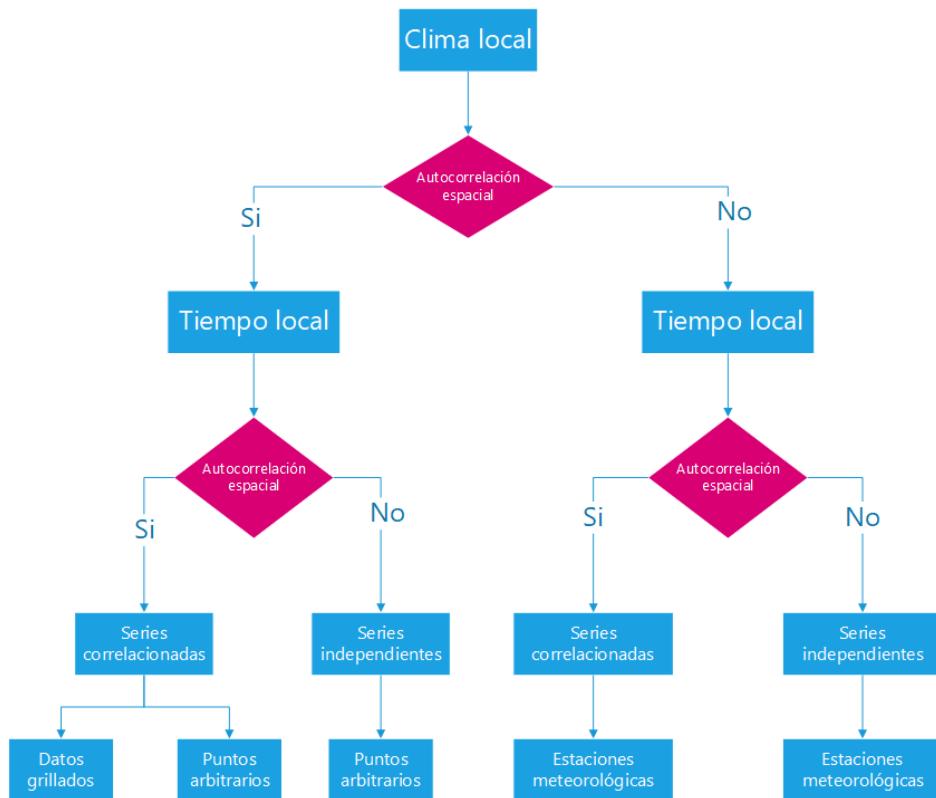


Figure 17: Diagrama de flujo del generador.

365 Los “caminos” asociados con cada variante del generador comienzan a partir de la extracción

³⁶⁶ de la información meteorológica y luego se bifurcan en función de como se considera la
³⁶⁷ autocorrelación espacial de las distintas estaciones meteorológicas usadas para entrenar los
³⁶⁸ modelos.

³⁶⁹ La rama izquierda de la Figura 17, corresponde al denominado “modelo espacial”. En esta
³⁷⁰ variante, la autocorrelación espacial forma parte del modelo que ajusta el clima local. Es
³⁷¹ decir, cada uno de los GAMs incorpora las coordenadas de cada estación meteorológica para
³⁷² modelar la autocorrelación espacial. Los datos de las estaciones meteorológicas de la red
³⁷³ se agrupan en una sola matriz y los procesos no se modelan de manera puntual, sino que
³⁷⁴ éstos varían en el espacio. Sólo se ajusta un modelo por variable meteorológica que agrupa
³⁷⁵ a todas las estaciones meteorológicas. Luego, al considerar la modelación del **tiempo local**,
³⁷⁶ nuevamente hay dos posibilidades. Si la dependencia espacial se considera, se generan series
³⁷⁷ completamente correlacionadas a través de campos gaussianos aleatorios por lo que el gener-
³⁷⁸ ador puede simular sobre una grilla regular o sobre puntos arbitrarios que no necesariamente
³⁷⁹ deben ser estaciones meteorológicas. En cambio, si la dependencia espacial no se considera
³⁸⁰ en la componente meteorológica, sólo pueden generarse series sobre puntos y no sobre una
³⁸¹ grilla porque los píxeles serían independientes y ésto no tiene sentido físico. En la rama
³⁸² derecha, en cambio, la autocorrelación espacial no forma parte de la *componente climática*
³⁸³ y corresponde al denominado “modelo local”. En esta variante se ajusta un modelo para
³⁸⁴ cada estación meteorológica. Luego, al simular la componente meteorológica, los caminos
³⁸⁵ vuelven a dividirse. Las estaciones pueden seguir siendo independientes y la componente
³⁸⁶ meteorológica puede ser estimada para cada estación sin considerar a sus vecinas. La otra
³⁸⁷ posibilidad consiste en agrupar todas las estaciones y estimar la componente meteorológica
³⁸⁸ considerando la existencia de autocorrelación espacial por lo que las estaciones más próximas
³⁸⁹ serían más similares entre sí comparadas con las más lejanas. En el paquete GAMWGEN las
³⁹⁰ funciones utilizadas para el “modelo espacial” tiene el sufijo **spatial** mientras que aquellas
³⁹¹ asociadas al “modelo local” tienen el sufijo **local**.

392 **5 Aplicación**

393 En esta sección se mostrarán ejemplos de aplicación del generador para generar distintos
394 tipos de series explicando las funciones necesarias y cada uno de los parámetros.

395 **5.1 Instalar paquetes necesarios**

396 El primer paso es comprobar que todos los paquetes necesarios estén instalados y si no es
397 así, descargarlos e instalarlos.

398 **5.2 Creación de directorios**

399 El paquete tiene precargados algunos ejemplos de aplicación con datos reales de estaciones
400 meteorológicas de la red del SISSA.

401 Para poder seguir este manual se deben crear directorios donde se guardarán los datos de
402 entrada y salida.

- 403 • /input_data: aquí se guardarán los datos meteorológicos y los metadatos de las esta-
404 ciones
- 405 • /output_data: aquí se guardarán los resultados de la simulación

406 Si estos directorios no existen, se crearán.

407 **5.3 Generación de series sintéticas sin autocorrelación espacial**

408 Este primer ejemplo consiste en la generación de series sintéticas para estaciones meteo-
409 rológicas sin considerar la dependencia espacial por lo que se utilizará el “modelo local”. Es
410 decir, se generarán series para las mismas estaciones que fueron utilizadas en el ajuste de
411 los distintos modelos. El ejemplo está dividido en tres, en una primera parte se ajustará el

412 modelo para generar series estacionarias y luego, en una segunda parte, se incluirán covari-
413 ables estacionales para producir series pseudohistóricas. Para ambos ejemplos los datos son
414 usados serán los mismos. Un tercer ejemplo mostrará como generar datos para más de una
415 estación meteorológica incluyendo la dependencia espacial al modelar el *tiempo local*.

416 **5.3.1 Creación de archivos de entrada**

417 El primer paso consiste en generar los set de datos de entrada que se descargaron al momento
418 de instalar el paquete del generador estocástico. Estos datos son sólo a título demostrativo,
419 si el usuario desea correr el modelo con sus propios datos deberá cambiar los objetos que se
420 generarán en esta sección por los suyos y colocarlos en la carpeta `input_data`.

421 Los archivos necesarios son:

- 422 • `stations.csv`
423 • `climate.csv`

424 Los datos meteorológicos se dividen en dos archivos separados: `stations.csv` y
425 `climate.csv`. Los nombres de los mismos no deben ser necesariamente iguales a los
426 usados aquí.

427 Los metadatos de las estaciones se alojan en el archivo `stations.csv`. Este archivo contiene
428 la información de las estaciones meteorológicas que serán usadas en el ajuste del modelo.

429 Las variables que deben ser incluidas en la tabla son:

- 430 • `station_id`: número único para cada estación meteorológica. La variable debe ser
431 de tipo *integer*
432 • `latitude`: latitud en grados decimales. La variable debe ser de tipo *double*
433 • `longitude`: longitud en grados decimales. La variable debe ser de tipo *double*

- ⁴³⁴ La tabla puede tener más variables pero sólo se necesitan las anteriores.
- ⁴³⁵ A continuación se muestran la primera fila del dataset y los tipos de datos de cada una de
- ⁴³⁶ las variables.

```
# Visualización de los metadatos de la estación
knitr::kable(head(stations), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")
```

x	y	station_id	nombre	lat_dec	lon_dec	elev	pais_id
5001614	6256841	87448	Villa Reynolds Aero	-33.7181	-65.3737	486	AR

- ⁴³⁷ El objeto **stations** debe ser convertido de *tibble* a *sf*. El sistema de referencia espacial debe ser planar. No es necesario un sistema de referencia espacial en particular, solamente las
- ⁴³⁹ coordenadas deben estar expresadas en metros.

```
# Se convierte el objeto stations a sf y se transforma su
# proyección de WGS 1984 a POSGAR Argentina Faja 5.
stations %<>%
  sf::st_as_sf(coords = c("lon_dec", "lat_dec"), crs = 4326) %>%
  sf::st_transform(crs = 22185)
```

- ⁴⁴⁰ La información climática se aloja en el archivo `climate.csv`. Este archivo contiene los datos
- ⁴⁴¹ de las estaciones meteorológicas que serán usadas en el ajuste del modelo. Las variables que
- ⁴⁴² deben ser incluidas en la tabla son:

- ⁴⁴³ • **date**: fecha del dato. La variable debe ser de tipo *date*
 - ⁴⁴⁴ • **station_id**: número único para cada estación meteorológica. La variable debe ser
- ⁴⁴⁵ de tipo *integer*

- 446 • `prcp`: datos diarios de precipitación La variable debe ser de tipo *double*
 447 • `tmax`: datos diarios de temperatura máxima. La variable debe ser de tipo *double*
 448 • `tmin`: datos diarios de temperatura mínima. La variable debe ser de tipo *double*

449 A continuación se muestran las primeras cinco filas del dataset y los tipos de datos de cada
 450 una de las variables.

```
# Visualización de los datos climáticos de la estación
```

```
knitr::kable(head(climate), "latex", booktabs = T) %>%
```

```
kableExtra::kable_styling(position = "center")
```

date	station_id	tmax	tmin	prcp
1961-01-01	87448	37.4	13.5	0.6
1961-01-02	87448	27.4	14.3	23.9
1961-01-03	87448	26.6	13.5	0.0
1961-01-04	87448	31.0	11.7	6.0
1961-01-05	87448	27.0	14.1	0.0
1961-01-06	87448	26.3	11.3	0.0

451 Los nombres de las variables son importantes y deben ser siempre los mismos ya que el
 452 modelo las reconocerá a partir de los mismos. Los nombres deben ser los siguientes:

- 453 • `date` : corresponde a la fecha del día en formato **Date**. El formato de la fecha para fa-
 454 cilitar el reconocimiento por parte de R es “YYYY-MM-DD”, es decir, el año expresado
 455 con cuatro dígitos y luego dos dígitos para el mes y dos para el día.
 456 • `station_id`: Identificador único de cada una de las estaciones. Debe ser un número
 457 entero.

- 458 • `tmax`: temperatura máxima diaria expresada en °C.
459 • `tmin`: temperatura mínima diaria expresada en °C.
460 • `prcp`: precipitación diaria expresada en mm.

461 El orden de las variables no es importante pero, como se mencionó, si se deben respetar los
462 nombres de cada una. En el caso de faltantes, no se utiliza ningún valor específico para los
463 NAs, sólo se debe dejar ese valor vacío. Este archivo tiene un formato largo, es decir, las
464 estaciones se deben colocar una debajo de la otra.

465 La generación de series sobre estaciones requiere de dos funciones básicas `local_fit` y
466 `local_simulate`. Independientemente de si se incluyen totales trimestrales en el modelo,
467 siempre se utilizan esas dos funciones.

468 **5.3.2 Series sintéticas estacionarias**

469 **5.3.2.1 Ajuste de los modelos** A continuación se mostrará como ajustar los cuatro
470 modelos estadísticos para una sola estación meteorológica para generar series estacionarias
471 donde cada realización es completamente independiente.

472 El ajuste del modelo local (en un punto) necesita de dos funciones: en una se define la
473 configuración general del modelo y con la segunda se corre el modelo propiamente dicho.
474 Amvas funciones tienen el sufijo `local` por delante del nombre.

475 Primero se crea un objeto con el control para el ajuste del simulador. Los argumentos son:

- 476 • `prcp_occurrence_threshold`: umbral de precipitación para un día lluvioso. La OMM
477 recomienda un umbral de 0.1 mm para considerar un día como lluvioso.
478 • `avbl_cores`: cantidad de núcleos disponibles para la parallelización.
479 • `planar_crs_in_metric_coords`: sistema de coordenadas planar.

Creación del objeto de control

```
control_fit <- gamwgen::local_fit_control(
```

```

prcp_occurrence_threshold = 0.1,
# Umbral para la definición de días húmedos
avbl_cores = 6,
# Cantidad de núcleos disponibles
planar_crs_in_metric_coords = 22185)
# Sistema de referencia espacial (en metros)

```

- ⁴⁸⁰ Luego se corre el ajuste para la estación meteorológica con la función `local_calibrate`.
⁴⁸¹ Los argumentos de la función son:

- ⁴⁸² • `climate`: datos meteorológicos observados para la estación
- ⁴⁸³ • `stations`: metadatos de las estaciones meteorológicas
- ⁴⁸⁴ • `seasonal_covariates`: datos agregados trimestrales. Si es NULL el ajuste será sin covariables y las series generadas serán estacionarias.
- ⁴⁸⁵ • `control`: objeto de control
- ⁴⁸⁶ • `verbose`: controla la impresión de mensajes en la consola. FALSE por defecto.

```

# Al correr la función se realiza el ajuste de los cuatro modelos
# para cada una de las estaciones. En este caso, por cuestiones de tiempo
# a cargar un objeto ya precalculado.
# Si el usuario desea correrlo deberá ver la nota anterior.

gamgen_fit <- gamwgen::local_calibrate(
  climate = climate,
  # Registro histórico de variables meteorológicas
  stations = stations,
  # Estaciones meteorológicas
  seasonal_covariates = NULL,
  # Totales trimestrales de precipitación

```

```

control = control_fit,
# Objeto de control
verbose = FALSE)

# Impresión de mensajes en la consola.

```

488 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```

# Se copia el archivo preajustado a nuestro directorio de trabajo

if (!fs::file_exists('input_data/local/fit_local_unconditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_unconditional.RData",
                           package = "gamwgen"),
                new_path = 'input_data/local/fit_local_unconditional.RData')
}

# Se carga el archivo recientemente creado

load('input_data/local/fit_local_unconditional.RData')

# Clase del objeto con el ajuste del generador

class(gamgen_fit)

489 ## [1] "gamwgen"

# Contenido del modelo

names(gamgen_fit)

490 ## [1] "control"                 "stations"                  "climate"
491 ## [4] "crs_used_to_fit"          "start_climatology"        "fitted_models"
492 ## [7] "models_data"              "models_residuals"         "statistics_threshold"
493 ## [10] "exec_times"

```

494 Dentro del objeto se guardan todo lo necesario para la simulación así como información
495 accesoria.

- 496 • **control**: copia de la configuración usada para calibrar el generador
497 • **stations**: estaciones meteorológicas utilizadas para la calibración
498 • **climate**: datos climáticos de cada uno de las estaciones
499 • **seasonal_covariates**: series temporales de totales trimestrales de precipitación y
500 medias trimestrales de temperaturas máxima y mínima.
501 • **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
502 • **start_climatology**: climatología diaria de cada una de las variables de entrada.
503 • **fitted_models**: modelos ajustados, uno para cada variable: temperaturas máxima y
504 mínima y ocurrencia y montos de precipitación.
505 • **models_data**: datos usados efectivamente usados para ajustar los modelos (sin NAs)
506 • **models_residuals**: residuos de cada uno de los modelos. Es decir, la diferencia entre
507 el valor ajustado por el modelo (clima local) y el valor observado en el día **i**
508 • **statistics_threshold**: umbrales de amplitud térmica diaria por mes. Si la amplitud
509 simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de
510 mantener la consistencia entre variables
511 • **exec_times**: tiempo de ejecución de cada una de las etapas del ajuste

512 Cada uno de los GAMs ajustados se almacenan en el objeto **gamgen_fit** y pueden ser
513 evaluados con la función **summary()**.

Visualización del GAM ajustado de temperatura máxima

summary(gamgen_fit\$fitted_models\$`87448`\$tmax_fit)

514 ##

515 ## Family: gaussian

```

516 ## Link function: identity
517 ##
518 ## Formula:
519 ##  $tmax \sim s(tmax\_prev, tmin\_prev, k = 50) + s(prcp\_occ, bs = "re") +$ 
520 ##  $s(prcp\_occ\_prev, bs = "re") + s(doy, bs = "cc", k = 30)$ 
521 ##
522 ## Parametric coefficients:
523 ## Estimate Std. Error t value Pr(>|t|)
524 ## (Intercept) 25.61343 0.03214 796.8 <2e-16 ***
525 ## ---
526 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
527 ##
528 ## Approximate significance of smooth terms:
529 ## edf Ref.df F p-value
530 ##  $s(tmax\_prev,tmin\_prev)$  30.2469 38.71 247.1 <2e-16 ***
531 ##  $s(prcp\_occ)$  0.9989 1.00 1005.8 <2e-16 ***
532 ##  $s(prcp\_occ\_prev)$  0.9995 1.00 2300.2 <2e-16 ***
533 ##  $s(doy)$  12.8918 28.00 158.9 <2e-16 ***
534 ## ---
535 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
536 ##
537 ## R-sq.(adj) = 0.71 Deviance explained = 71.1%
538 ## fREML = 58955 Scale est. = 14.138 n = 21466

```

539 La función `summary` permite analizar los resultados del ajuste de cada uno de los modelos.
 540 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la parte
 541 superior bajo el apartado `Formula` y la significancia de cada uno de los términos del modelo
 542 en la tabla inmediatamente inferior. Se puede observar que todos los términos son altamente

543 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la
544 varianza explicada por el modelo y el valor de R-ajustado.

545 El paquete `gratia` (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
546 de manera gráfica. Esta función toma los diagnósticos por defecto de la función `gam.check`
547 del paquete `mgcv` (Wood, Simon (2015)) y los convierte en gráficos de la librería `ggplot2`
548 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La Figura 18 está dividida en
549 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
550 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
551 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
552 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los
553 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
554 en los residuos indicaría que el modelo no ha explicado algún componente importante de
555 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
556 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
557 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
558 ajustados y observados.

Diagnósticos gráficos del modelo

```
gratia::appraise(gamgen_fit$fitted_models$`87448`$tmax_fit) +  
  ggplot2::theme_bw()
```

559 Estos diagnóstico exploratorios puede aplicarse a cada uno de los modelos ajustados por la
560 función `local_calibrate`.

561 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
562 series sintéticas para la o las estaciones usadas para calibrar el generador.

563 **5.3.2.2 Generación de series** La generación de series sigue la misma estructura ante-
564 rior, una función para configurar la generación y otra que realiza la generación propiamente

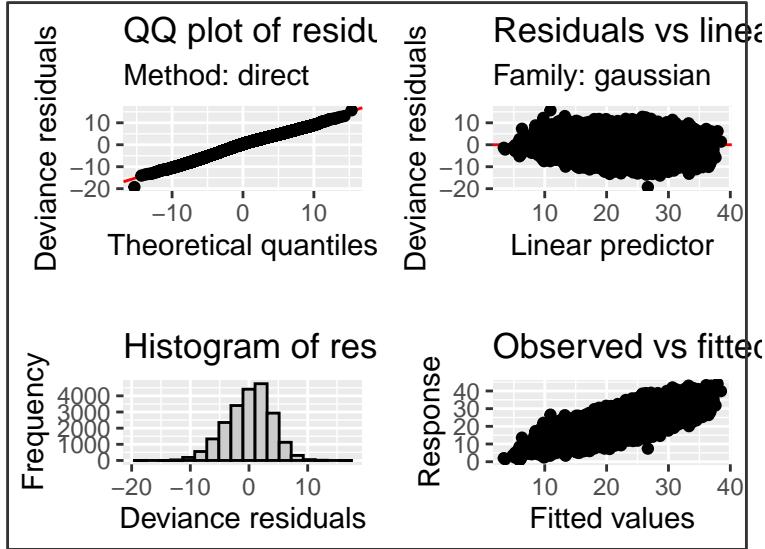


Figure 18: Diagnóstico del GAM de temperatura máxima de la estación 87448.

565 dicha.

566 Los argumentos de la función de control son:

- 567 • **nsim**: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o
568 igual a 1.
- 569 • **seed**: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo
570 porque se guarda junto a los resultados.
- 571 • **avbl_cores**: cantidad de núcleos disponibles para la paralelización.
- 572 • **use_spatially_correlated_noise**: utilizar la generación estocástica espacialmente
573 correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron
574 más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible
575 calcular los variogramas necesarios para la generación de los campos aleatorios. Se
576 debe introducir un **boolean** (TRUE or FALSE).
- 577 • **use_temporary_files_to_save_ram**: si se simulan muchas realizaciones o los recursos
578 informáticos son escasos, esta opción permite guardar los resultados de cada una de las
579 realizaciones en el disco liberando memoria RAM que quedará disponible para generar
580 nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno

- 581 único. Se debe introducir un **boolean** (TRUE or FALSE).
- 582 • **use_temporary_files_to_save_ram**: esta opción permite eliminar los archivos tem-
- 583 porales creados para ahorrar RAM luego de terminar la generación de todas las simu-
- 584 laciones. Se debe introducir un **boolean** (TRUE or FALSE).

```
# Se crea el objeto de control de la simulación
control_sim <- gamwgen::local_simulation_control(
  nsim = 10,
  # Cantidad de simulaciones a realizar
  seed = 1234,
  # Semilla para que los resultados sean reproducibles
  avbl_cores = 6,
  # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = FALSE,
  # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = FALSE,
  # Guardar resultados intermedios para ahorrar RAM
  remove_temp_files_used_to_save_ram = TRUE)
  # Borrar los resultados intermedios creados anteriormente
```

- 585 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de
- 586 simulación son:

- 587 • **model**: objeto con el resultado de la función `local_calibrate()`
- 588 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simu-
- 589 lar. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es
- 590 necesario que todas estén presentes, se pueden generar series solo sobre algunas de
- 591 ellas.

- 592 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron
 593 covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria.
 594 Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de
 595 covariables, ni tampoco posterior. Se debe introducir una fecha en formato **date**
 596
 597 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron co-
 598 variables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso
 599 contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables,
 600 tampoco puede ser posterior. Se debe introducir una fecha en formato **date**
 601
 602 • **control**: objeto de control creado con la función **control_sim()**.
 603
 604 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales
 605 como intermedios.
 606
 607 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad,
 608 el archivo generado es un archivo de texto en formato separado por comas (.csv)
 609
 610 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con co-
 611 variables, la generación también debe realizarse con ellas. Caso contrario se producirá
 612 un error. Se debe introducir un data frame con los valores agregados para las tres
 613 variables (precipitación y temperaturas máxima y mínima) pero no necesariamente
 deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de
 algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar
 estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma
 longitud que el período a generar
 614
 615 • **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```
# Al correr la función se realiza la generación de series para
# cada una de las estaciones.

# En este caso, por cuestiones de tiempo, vamos a cargar un objeto
# con los resultados de la simulación

simulated_climate <- gamwgen::local_simulation(
```

```

model = gamgen_fit,
# Objeto con los resultados del ajuste
simulation_locations = stations,
# Estaciones para las cuales simular
start_date = as.Date('2019-01-01'),
# Fecha de comienzo de las simulaciones
end_date = as.Date('2019-12-31'),
# Fecha de fin de las simulaciones
control = control_sim,
# Objeto con la configuración
output_folder = getwd(),
# Directorio donde se guardarán los resultados
output_filename = 'simulations.csv',
# Nombre del archivo de salida
seasonal_covariates = NULL,
# Covariables estacionales
verbose = FALSE)
# Impresión de mensajes en la consola

```

- 614 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
 615 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```

# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('output_data/simulated_local_unconditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "simulated_local_unconditional.RData",
                           package = "gamwgen"),
               new_path = 'output_data/simulated_local_unconditional.RData')
}

```

```

# Se carga el archivo recientemente creado
load('output_data/simulated_local_unconditional.RData')

# Clase del objeto con el ajuste del generador
class(simulated_climate)

616 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo
names(simulated_climate)

617 ## [1] "nsim"
618 ## [2] "seed"
619 ## [3] "realizations_seeds"
620 ## [4] "simulation_points"
621 ## [5] "output_file_with_results"
622 ## [6] "output_file_fomart"
623 ## [7] "rdata_file_with_fitted_stations_and_climate"
624 ## [8] "exec_times"

```

625 La lista contiene los siguientes objetos:

- 626 • **nsim**: cantidad de realizaciones.
- 627 • **seed**: semilla general para toda la generación. Corresponde a la que se incluye en la función de control.
- 628 • **realization_seeds**: semillas para cada una de las realizaciones. Esto permite replicar los resultados.
- 629 • **simulation_points**: puntos donde se generaron las series sintéticas.
- 630 • **output_file_with_results**: nombre del archivo con los resultados.

- 633 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.
- 634 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
- 635 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
- 636 metadatos de cada uno de esos puntos.
- 637 • `exec_times`: tiempo de ejecución de la generación.

638 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

639 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones

640 que forman parte de `local_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen

641 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los

642 parámetros.

643 En la tabla anterior se muestran los parámetros necesarios para generar el tiempo local de

644 las temperaturas máxima y mínima.

```
# Función para la generación de los parámetros del
# modelo de tiempo local

gen_noise_params <- gamwgen:::generate_residuals_statistics(
  models_residuals = gamgen_fit$models_residuals)

# Visualización de los parámetros

knitr::kable(head(gen_noise_params), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center",
    latex_options = c("striped", "scale_down"))
```

station_id	type_day	month	sd.tmax_residuals	sd.tmin_residuals	mean.tmax_residuals	mean.tmin_residuals	cov.residuals	var.tmax_residuals	var.tmin_residuals
87448	Wet	1	3.254469	2.543695	0.8526532	-0.3860514	1.0779804	12.389767	4.907709
87448	Dry	1	3.254469	2.543695	-0.3494070	0.2137263	1.6070642	9.379906	7.045536
87448	Wet	2	3.352673	2.589353	0.7087247	-0.2850597	0.8544476	13.069030	4.397535
87448	Dry	2	3.352673	2.589353	-0.3295625	0.0035795	1.5816505	10.215926	7.603715
87448	Dry	3	3.444108	2.678806	0.0577372	0.0427601	1.6112722	10.910244	7.743939
87448	Wet	3	3.444108	2.678806	-0.2366143	-0.0993789	1.8594848	14.323711	5.682934

- `station_id`: número único que identifica a cada estación meteorológica.
- `type`: tipo de día **lluvioso (Wet)** o **seco (Dry)**.
- `month`: número de mes para los que se calculan los parámetros
- `sd.tmax_residuals`: desvío estándar de los residuos del modelo de temperatura máxima.
- `sd.tmin_residuals`: desvío estándar de los residuos del modelo de temperatura mínima.
- `mean.tmax_residuals`: media de los residuos del modelo de temperatura máxima.
- `mean.tmin_residuals`: media de los residuos del modelo de temperatura mínima *
- `cov.residuals`: covarianza de los residuos.
- `var.tmax_residuals`: covarianza de los residuos del modelo de temperatura máxima.
- `var.tmin_residuals`: covarianza de los residuos del modelo de temperatura mínima.

Los parámetros para cada uno de los meses permiten generar valores de tiempo local para las dos temperaturas a partir de una distribución normal multivariada.

A continuación se muestra en la Figura 19 un ejemplo para el año 2019 sólo para los días secos.

```
# Fechas para la generación de los campos de tiempo local
fechas <- data.frame(
  date = seq(as.Date('2019-01-01'), as.Date('2019-12-31'), 'days')) %>%
  dplyr::mutate(dia = lubridate::day(date),
  mes = lubridate::month(date))
```

Ejemplo de generación de tiempo local de temperatura

para días secos para el mes de enero

```
temp_dry <- purrr::map2_dfr(
```

```
.x = fechas$dia,
```

```

.y = fechas$mes,
.f = function(dia, mes) {

  result_temp_dry <- control_sim$temperature_noise_generating_function(
    simulation_points = stations %>%
      dplyr::filter(., station_id == '87448'),
    gen_noise_params = gen_noise_params,
    month_number = mes,
    selector = 'tmax_dry',
    seed = NULL)

  result_temp_dry <- result_temp_dry %>%
    dplyr::mutate(dia = dia,
      mes = mes)

}

) %>%
  sf::st_set_geometry(NULL) %>%
  dplyr::mutate(date = as.Date(paste0('2019-', mes, '-', dia))) %>%
  dplyr::select(-mes, -dia)

# Visualización de los campos

ggplot2::ggplot(data = temp_dry %>%
  tidyr::gather(residuo, valor, -date),
  ggplot2::aes(x = date, y = valor, color = residuo)) +
  ggplot2::scale_y_continuous(limits = c(-15, 15),
    breaks = seq(-15, 15, 3),

```

```

          name = 'Tiempo local') +
ggplot2::scale_x_date(name = 'Días') +
ggplot2::scale_color_manual(values=c("DarkOrange", "Steelblue"),
                           labels = c("Temp. Máx.", "Temp. Min.")) +
ggplot2::geom_line() +
ggplot2::theme_bw() +
ggplot2::theme(legend.position="bottom",
              legend.title = ggplot2::element_blank())

```

661 ## Warning: attributes are not identical across measure variables;
 662 ## they will be dropped

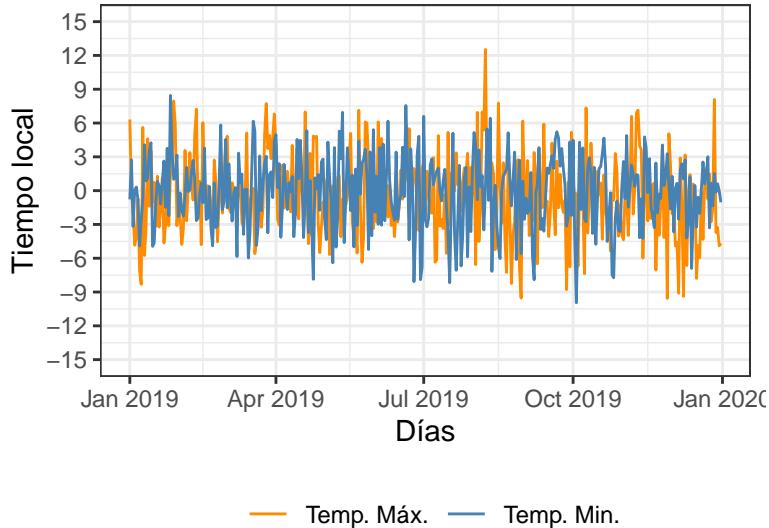


Figure 19: Tiempo local de temperatura para enero de 2019 de la estación 87448.

663 La línea naranja corresponde a la serie para temperaturas máximas y la azul para temper-
 664 aturas mínimas.

```

# Se carga el set de datos simulados
simulated_climate <- readr::read_csv(

```

```

here::here('output_data/local/simulated_local_unconditional.csv'))

# Primeras filas del objeto de salidas

knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center",
                            latex_options = c("striped", "scale_down"))

```

realization	station_id	point_id	longitude	latitude	date	tmax	tmin	prcp
1	87448	1	5001636	6256577	2019-01-01	27.19266	5.464899	0
1	87448	1	5001636	6256577	2019-01-02	30.57296	10.266542	0
1	87448	1	5001636	6256577	2019-01-03	33.36446	15.180860	0
1	87448	1	5001636	6256577	2019-01-04	34.05627	15.473379	0
1	87448	1	5001636	6256577	2019-01-05	32.69240	15.270145	0
1	87448	1	5001636	6256577	2019-01-06	30.56683	14.797530	0

665 El resultado de la generación es un archivo .csv que contiene la siguiente información:

- 666 • **realization**: número de realización. Es un valor entero entre 1 y la cantidad de
667 realizaciones definida por el usuario.
- 668 • **station_id**: número único de identificación de la estación meteorológica o del punto
669 arbitrario.
- 670 • **date**: fechas de cada uno de los días de la simulación.
- 671 • **tmax**: valores de temperatura máxima generada expresada en °C.
- 672 • **tmin**: valores de temperatura mínima generada expresada en °C.
- 673 • **prcp**: valores de precipitación diaria generada expresada en mm.

674 La Figura 20 muestra un ejemplo de las series de temperaturas máximas y mínimas gener-
675 adas.

```

# Grafico de temperatura máxima

tmax_plot <- ggplot2::ggplot() +
  ggplot2::geom_line(data = simulated_climate,
    ggplot2::aes(x = date, y = tmax, color = realization),
    alpha = 0.5, color = 'DarkOrange') +
  ggplot2::geom_line(data = climate %>%
    dplyr::filter(date > as.Date('2019-01-01')),
    ggplot2::aes(x = date, y = tmax)) +
  ggplot2::labs(x = 'Fecha', y = 'Temperatura máxima [°C]') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = "none")

# Grafico de temperatura mínima

tmin_plot <- ggplot2::ggplot() +
  ggplot2::geom_line(data = simulated_climate,
    ggplot2::aes(x = date, y = tmin, color = realization),
    alpha = 0.5, color = 'Steelblue') +
  ggplot2::geom_line(data = climate %>%
    dplyr::filter(date > as.Date('2019-01-01')),
    ggplot2::aes(x = date, y = tmin)) +
  ggplot2::labs(x = 'Fecha', y = 'Temperatura mínima [°C]') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = "none")

# Combinación de ambos gráficos en un solo panel

cowplot::ggdraw() +
  cowplot::draw_plot(tmax_plot, x = 0, y = 0.5, width = 1, height = .5) +

```

```
cowplot::draw_plot(tmin_plot, x = 0, y = 0, width = 1, height = .5)
```

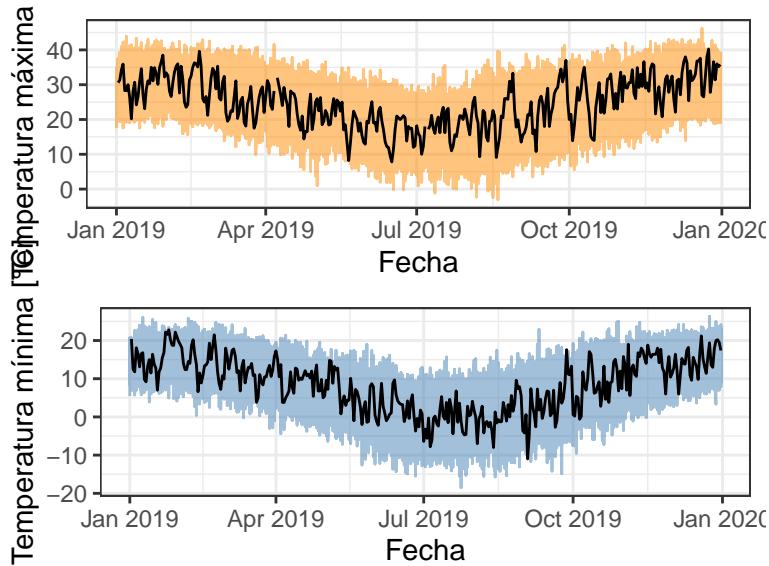


Figure 20: Temperatura máxima (superior) y mínima (inferior) generada para enero de 2019 de la estación 87448.

676 En el panel superior se muestra la temperatura máxima observada (negra) y las temperaturas
677 sintéticas (naranja) para el año 2019. En el inferior se muestra la temperatura mínima diaria
678 observada (negra) y cada una de las realizaciones (azul).

679 5.3.3 Series sintéticas pseudohistóricas

680 **5.3.3.1 Ajuste de los modelos** A continuación se mostrará como ajustar los cuatro
681 modelos estadísticos para una sola estación meteorológica para generar series **pseudo-**
682 **históricas** donde cada realización copia las variaciones de baja frecuencia e la serie ob-
683 servada. El ajuste del modelo local (en un punto) necesita de dos funciones: en una se define
684 la configuración general del modelo y con la segunda se corre el modelo propiamente dicho.

685 Primero se crea un objeto con el control para el ajuste del simulador. Los argumentos son:

- 686 • `prcp_occurrence_threshold`: umbral de precipitación para un día lluvioso. La OMM
687 recomienda un umbral de 0.1 mm para considerar un día como lluvioso.

- 688 • `avbl_cores`: cantidad de núcleos disponibles para la paralelización.
- 689 • `planar_crs_in_metric_coords`: sistema de coordenadas planar.

Creación de un objeto de control del ajuste.

```
control_fit <- gamwgen::local_fit_control(
  prcp_occurrence_threshold = 0.1,
  # Umbral para la definición de días húmedos
  avbl_cores = 6,
  # Cantidad de núcleos disponibles
  planar_crs_in_metric_coords = 22185)
# Sistema de referencia espacial (en metros)
```

690 Al tratarse de un modelo que ajusta condicionado por la variabilidad de baja frecuencia
 691 preexistente en los datos observados es necesario la agregación de las variables diarias en to-
 692 tales trimestrales de precipitación y medias trimestrales de temperaturas máxima y mínima.
 693 Esta operación puede realizarse con la función `summarise_seasonal_climate` incluida en
 694 el paquete. Esta función, además de agregar los datos, permite la imputación de faltantes.
 695 Se toleran una cierta cantidad que puede ser determinada por el usuario. El método de
 696 imputación utilizado es el `imputePCA()` de la librería `missMDA`.

Agregación de valores diarios

```
seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes =
```

Se muestran las primeras cinco filas

```
knitr::kable(head(seasonal_covariates), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")
```

station_id	year	season	seasonal_prcp	seasonal_tmax	seasonal_tmin
87448	1961	1	273.7	30.98764	14.513483
87448	1961	2	236.0	24.47473	8.467033
87448	1961	3	11.3	19.04565	1.345652
87448	1961	4	234.5	24.66235	7.761177
87448	1962	1	402.4	29.92333	14.245556
87448	1962	2	187.2	24.30440	7.986813

697 Cabe mencionar que con esta función las valores se agregan por trimestre considerando la
698 siguiente definición:

- 699 • Verano: Diciembre, Enero y Febrero
700 • Otoño: Marzo, Abril y Mayo
701 • Invierno: Junio, Julio y Agosto
702 • Primavera: Septiembre, Octubre y Noviembre

703 Los valores también se podrían agregar siguiendo otra definición de estaciones pero en ese
704 caso, el usuario debería hacerlo por su cuenta. Algunas funciones útiles para hacerlo son
705 las disponibles en el paquete `lubridate` como `quarter()` que permite definir el mes de
706 comienzo de los trimestres. Para estas variables los nombres también son importantes por
707 lo que deben respetarse los mostrados anteriormente.

708 La siguiente Figura 21 muestra los totales trimestrales de precipitación para la estación
709 meteorológica del ejemplo.

```
# Gráfico de precipitación acumulada por trimestre
```

```
ggplot2::ggplot(data = seasonal_covariates %>%
  dplyr::mutate(season = factor(season,
```

```

      labels = c('Verano', 'Otoño', 'Invierno')

ggplot2::aes(x = year, y = seasonal_prcp, group = 1)) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(se = FALSE, span = 0.5) +
  ggplot2::facet_wrap(~season, scales = 'free') +
  ggplot2::labs(x = 'Años', y = 'Precipitación acumulada [mm]') +
  ggplot2::theme_bw()

710 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

```

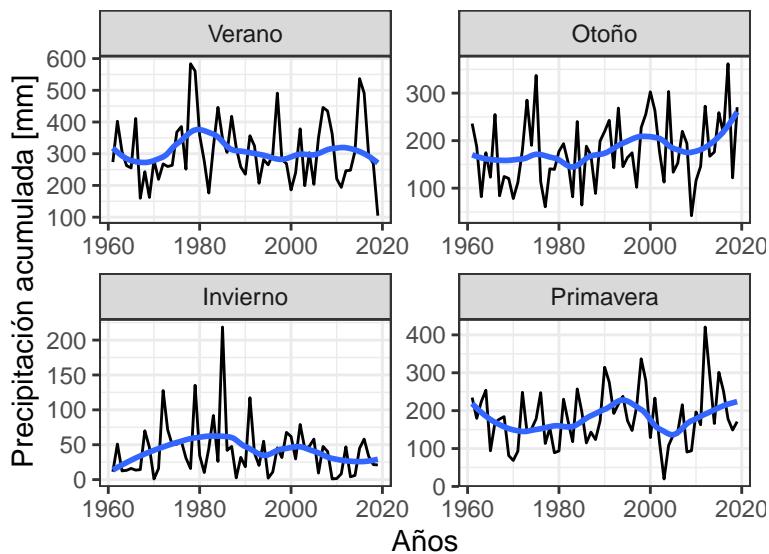


Figure 21: Precipitación acumulada por trimestre para la estación 87448.

711 Cada uno de los paneles muestra la precipitación acumulada por trimestre junto a una
 712 regresión local (loess) en azul. Al incluir estos totales en el modelo, las series generadas
 713 tenderán a seguir la variabilidad observada, es decir, años secos generarán realización con
 714 valores inferiores a la media y viceversa.

715 Luego se corre el ajuste para la estación meteorológica con la función `local_calibrate`.

716 Los argumentos de la función son:

- 717 • `climate`: datos meteorológicos observados para la estación

- 718 • **stations**: metadatos de las estaciones meteorológicas
- 719 • **seasonal_covariates**: datos agregados trimestrales. Si es NULL el ajuste será sin
720 covariables y las series generadas serán estacionarias.
- 721 • **control**: objeto de control
- 722 • **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```
# Al correr la función se realiza el ajuste de los cuatro modelos para
# cada una de las estaciones. En este caso, por cuestiones de tiempo
# a cargar un objeto ya precalculado.

# Si el usuario desea correrlo deberá ver la nota anterior.

gamgen_fit <- gamwgen::local_calibrate(
  climate = climate,
  # Registro histórico de variables meteorológicas
  stations = stations,
  # Estaciones meteorológicas
  seasonal_covariates = seasonal_covariates,
  # Totales trimestrales de precipitación
  control = control_fit,
  # Objeto de control
  verbose = FALSE)

# Impresión de mensajes en la consola.
```

- 723 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```
# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/local/fit_local_conditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_conditional.RData",
    package = "gamwgen"),
```

```

    new_path = 'input_data/local/fit_local_conditional.RData')

}

# Se carga el archivo recientemente creado

load('input_data/local/fit_local_conditional.RData')

# Clase del objeto con el ajuste del generador

class(gamgen_fit)

724 ## [1] "gamwgen"

# Contenido del modelo

names(gamgen_fit)

725 ## [1] "control"           "stations"          "climate"
726 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
727 ## [7] "fitted_models"       "models_data"        "models_residuals"
728 ## [10] "statistics_threshold" "exec_times"

```

729 Dentro del objeto se guardan todo lo necesario para la simulación así como información
 730 accesoria.

- 731 • **control**: copia de la configuración usada para calibrar el generador
- 732 • **stations**: estaciones meteorológicas utilizadas para la calibración
- 733 • **climate**: datos climáticos de cada uno de las estaciones
- 734 • **seasonal_covariates**: series temporales de totales trimestrales de precipitación y
 735 medias trimestrales de temperaturas máxima y mínima.
- 736 • **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
- 737 • **start_climatology**: climatología diaria de cada una de las variables de entrada.

```

738     • fitted_models: modelos ajustados, uno para cada variable: temperaturas máxima y
739         mínima y ocurrencia y montos de precipitación.
740
741     • models_data: datos usados efectivamente usados para ajustar los modelos (sin NAs)
742
743     • models_residuals: residuos de cada uno de los modelos. Es decir, la diferencia entre
744         el valor ajustado por el modelo (clima local) y el valor observado en el día i
745
746     • statistics_threshold: umbrales de amplitud térmica diaria por mes. Si la amplitud
747         simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de
748         mantener la consistencia entre variables
749
750     • exec_times: tiempo de ejecución de cada una de las etapas del ajuste

747 Cada uno de los GAMs ajustados se almacenan en el objeto gamgen_fit y pueden ser
748 evaluados con la función summary().

```

```

# Visualización del GAM ajustado de temperatura máxima.

summary(gamgen_fit$fitted_models$`tmax_fit`)

749 ##
750 ## Family: gaussian
751 ## Link function: identity
752 ##
753 ## Formula:
754 ## tmax ~ s(tmax_prev, tmin_prev, k = 50) + s(prcp_occ, bs = "re") +
755 ##      s(prcp_occ_prev, bs = "re") + s(doy, bs = "cc", k = 30) +
756 ##      s(SX1, SN1, k = 20) + s(SX2, SN2, k = 20) + s(SX3, SN3, k = 20) +
757 ##      s(SX4, SN4, k = 20)
758 ##
759 ## Parametric coefficients:
760 ##                               Estimate Std. Error t value Pr(>|t|)

```

```

761 ## (Intercept) 25.62116    0.03189   803.4   <2e-16 ***
762 ##
763 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
764 ##
765 ## Approximate significance of smooth terms:
766 ##                               edf Ref.df      F p-value
767 ## s(tmax_prev,tmin_prev) 29.9069 38.405 225.74 <2e-16 ***
768 ## s(prcp_occ)            0.9989  1.000  930.30 <2e-16 ***
769 ## s(prcp_occ_prev)       0.9995  1.000 2248.11 <2e-16 ***
770 ## s(doy)                 12.6336 28.000   73.12 <2e-16 ***
771 ## s(SX1,SN1)             3.0997  3.689   55.32 <2e-16 ***
772 ## s(SX2,SN2)             2.0001  2.000   89.25 <2e-16 ***
773 ## s(SX3,SN3)             4.4053  5.853   35.79 <2e-16 ***
774 ## s(SX4,SN4)             2.0001  2.000   86.85 <2e-16 ***
775 ##
776 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
777 ##
778 ## R-sq.(adj) =  0.713  Deviance explained = 71.4%
779 ## fREML =  58829  Scale est. = 13.964    n = 21466

```

780 La función **summary** permite analizar los resultados del ajuste de cada uno de los modelos.
781 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la
782 parte superior bajo el apartado **Formula** y la significancia de cada uno de los términos
783 del modelo en la tabla inmediatamente inferior. En este configuración, al incluirle variables
784 estacionales, se incorporan nuevos términos el modelo como SX1, SX2, SX3, SX4 y SN1, SN2,
785 SN3, SN4, que corresponden a variables dummy de las medias trimestrales de temperatura
786 máxima y mínima, respectivamente. Se puede observar que todos los términos son altamente
787 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la

788 varianza explicada por el modelo y el valor de R-ajustado.
789 El paquete **gratia** (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
790 de manera gráfica. Esta función toma los diagnósticos por defecto de la función **gam.check**
791 del paquete **mgcv** (Wood, Simon (2015)) y los convierte en gráficos de la librería **ggplot2**
792 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La Figura 22 está dividida en
793 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
794 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
795 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
796 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los
797 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
798 en los residuos indicaría que el modelo no ha explicado algún componente importante de
799 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
800 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
801 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
802 ajustados y observados.

Diagnósticos gráficos del modelo

```
gratia::appraise(gamgen_fit$fitted_models`^87448`$tmax_fit) +  
  ggplot2::theme_bw()
```

803 Estos diagnósticos exploratorios puede aplicarse a cada uno de los modelos ajustados por la
804 función **local_calibrate**.

805 Con la creación del objeto **gamgen_fit** finaliza el proceso de ajuste y ya se pueden generar
806 series sintéticas para la o las estaciones usadas para calibrar el generador.

807 **5.3.3.2 Generación de series** La generación de series sigue la misma estructura ante-
808 rior, una función para configurar la generación y otra que realiza la generación propiamente
809 dicha.

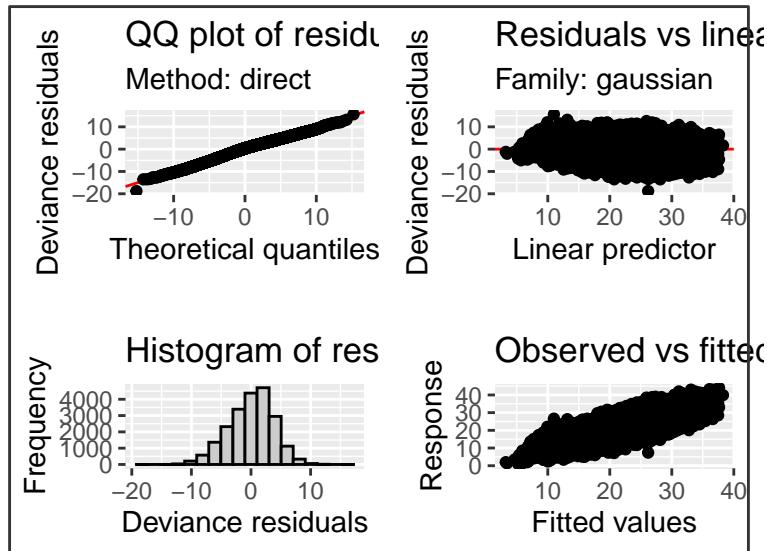


Figure 22: Diagnóstico del GAM de temperatura máxima de la estación 87448.

810 Los argumentos de la función de control son:

- 811 • **nsim**: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o
812 igual a 1.
- 813 • **seed**: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo
814 porque se guarda junto a los resultados.
- 815 • **avbl_cores**: cantidad de núcleos disponibles para la paralelización.
- 816 • **use_spatially_correlated_noise**: utilizar la generación estocástica espacialmente
817 correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron
818 más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible
819 calcular los variogramas necesarios para la generación de los campos aleatorios. Se
820 debe introducir un **boolean** (TRUE or FALSE).
- 821 • **use_temporary_files_to_save_ram**: si se simulan muchas realizaciones o los recursos
822 informáticos son escasos, esta opción permite guardar los resultados de cada una de las
823 realizaciones en el disco liberando memoria RAM que quedará disponible para generar
824 nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno
825 único. Se debe introducir un **boolean** (TRUE or FALSE).

- 826 • `use_temporary_files_to_save_ram`: esta opción permite eliminar los archivos temporales creados para ahorrar RAM luego de terminar la generación de todas las simulaciones. Se debe introducir un **boolean** (TRUE or FALSE).
- 827
- 828

```
# Creación del objeto de control de la simulación
control_sim <- gamwgen::local_simulation_control(
  nsim = 10,
  # Cantidad de simulaciones a realizar
  seed = 1234,
  # Semilla para que los resultados sean reproducibles
  avbl_cores = 6,
  # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = FALSE,
  # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = FALSE,
  # Guardar resultados intermedios para ahorrar RAM
  remove_temp_files_used_to_save_ram = TRUE)
# Borrar los resultados intermedios creados anteriormente
```

829 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de
 830 simulación son:

- 831 • `model`: objeto con el resultado de la función `local_calibrate()`
- 832 • `simulation_locations`: objeto tipo `sf` con la ubicación de las estaciones a simular. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es
 833 necesario que todas estén presentes, se pueden generar series solo sobre algunas de
 834 ellas.
- 835
- 836 • `start_date`: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron
 837 covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria.

Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, ni tampoco posterior. Se debe introducir una fecha en formato **date**

- **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron covariables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, tampoco puede ser posterior. Se debe introducir una fecha en formato **date**
- **control**: objeto de control creado con la función **control_sim()**.
- **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales como intermedios.
- **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad, el archivo generado es un archivo de texto en formato separado por comas (.csv)
- **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con covariables, la generación también debe realizarse con ellas. Caso contrario se producirá un error. Se debe introducir un data frame con los valores agregados para las tres variables (precipitación y temperaturas máxima y mínima) pero no necesariamente deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma longitud que el período a generar
- **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```

# Al correr la función se realiza la generación de series para cada una de las estacio
# En este caso, por cuestiones de tiempo, vamos a cargar un objeto
# con los resultados de la simulación

simulated_climate <- gamwgen::local_simulation(
  model = gamgen_fit,
  # Objeto con los resultados del ajuste
  simulation_locations = stations,

```

```

# Estaciones para las cuales simular
start_date = as.Date('2010-01-01'),
# Fecha de comienzo de las simulaciones
end_date = as.Date('2019-12-31'),
# Fecha de fin de las simulaciones
control = control_sim,
# Objeto con la configuración
output_folder = getwd(),
# Directorio donde se guardarán los resultados
output_filename = 'simulations.csv',
# Nombre del archivo de salida
seasonal_covariates = seasonal_covariates,
# Covariables estacionales
verbose = FALSE)
# Impresión de mensajes en la consola

```

- 858 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
 859 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```

# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('output_data/local/simulated_local_conditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "simulated_local_conditional.RData",
                           package = "gamwgen"),
               new_path = 'output_data/local/simulated_local_conditional.RData')
}

# Se carga el archivo recientemente creado
load('output_data/local/simulated_local_conditional.RData')

```

```

# Clase del objeto con el ajuste del generador

class(simulated_climate)

860 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo

names(simulated_climate)

861 ## [1] "nsim"
862 ## [2] "seed"
863 ## [3] "realizations_seeds"
864 ## [4] "simulation_points"
865 ## [5] "output_file_with_results"
866 ## [6] "output_file_fomart"
867 ## [7] "rdata_file_with_fitted_stations_and_climate"
868 ## [8] "exec_times"

```

869 La lista contiene los siguientes objetos:

- 870 • **nsim**: cantidad de realizaciones.
- 871 • **seed**: semilla general para toda la generación. Corresponde a la que se incluye en la
872 función de control.
- 873 • **realization_seeds**: semillas para cada una de las realizaciones. Esto permite replicar
874 los resultados.
- 875 • **simulation_points**: puntos donde se generaron las series sintéticas.
- 876 • **output_file_with_results**: nombre del archivo con los resultados.
- 877 • **output_file_format**: tipo de archivo de salida, en este caso .csv.
- 878 • **rdata_file_with_fitted_stations_and_climate**: archivo .RData con los datos me-
879 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
880 metadatos de cada uno de esos puntos.

- 881 • `exec_times`: tiempo de ejecución de la generación.

882 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

```
# Se carga el set de datos simulados
simulated_climate <- readr::read_csv(
  here::here('output_data/local/simulated_local_conditional.csv'))

# Primeras filas del objeto de salidas
knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

realization	station_id	point_id	longitude	latitude	date	tmax	tmin	prcp
1	87448	1	5001636	6256577	2010-01-01	28.05965	17.69135	7.534622
1	87448	1	5001636	6256577	2010-01-02	27.32783	16.20019	0.000000
1	87448	1	5001636	6256577	2010-01-03	35.01025	13.35766	0.000000
1	87448	1	5001636	6256577	2010-01-04	34.18990	18.04710	0.000000
1	87448	1	5001636	6256577	2010-01-05	30.59661	20.62692	0.000000
1	87448	1	5001636	6256577	2010-01-06	37.15162	17.44571	0.000000

883 Al utilizar totales trimestrales de precipitación y medias trimestrales de temperaturas máx-
884 ima y mínima, las series generadas capturan las variaciones de baja frecuencia que se observan
885 en la serie histórica. A continuación se muestra una Figura que ilustra lo anterior para la
886 temperatura máxima.

```
ggplot2::ggplot() +
  ggplot2::geom_boxplot(data = simulated_climate %>%
    dplyr::mutate(month = lubridate::month(date),
```

```

    year = lubridate::year(date),
    date = as.Date(paste0(year, '-1', month, '-1', 15L))),
ggplot2::aes(x = date, y = tmax,
             group = date, fill = 'DarkOrange'), alpha = 0.1) +
ggplot2::geom_line(data = climate %>%
dplyr::filter(date > as.Date('2010-01-01')) %>%
dplyr::mutate(month = lubridate::month(date),
              year = lubridate::year(date)) %>%
dplyr::group_by(year, month) %>%
dplyr::summarise(tmax = median(tmax, na.rm = TRUE)) %>%
dplyr::mutate(date = as.Date(paste0(year, '-1', month, '-1', 15L))),
ggplot2::aes(x = date, y = tmax, group = 1, color = 'DarkOrange')) +
ggplot2::theme_bw() +
ggplot2::labs(x = 'Fecha', y = 'Temperatura máxima [°C]') +
ggplot2::scale_x_date(breaks = '1 year',
                      labels = scales::date_format("%m-%y")) +
ggplot2::scale_fill_discrete(name = "", labels = c("Simulado")) +
ggplot2::scale_color_discrete(name = "", labels = c("Observado")) +
ggplot2::theme(legend.position = 'bottom')

```

- 887 Las cajas corresponde a las distintas realizaciones agregadas a escala mensual y la línea
 888 naranja corresponde a la temperatura media mensual calculada para los años 2010 a 2019.
 889 Se observa como las cajas suben y bajan al ritmo de la media observada y como capturan
 890 los pequeños cambios que ocurren en un año específico pero no en otros.

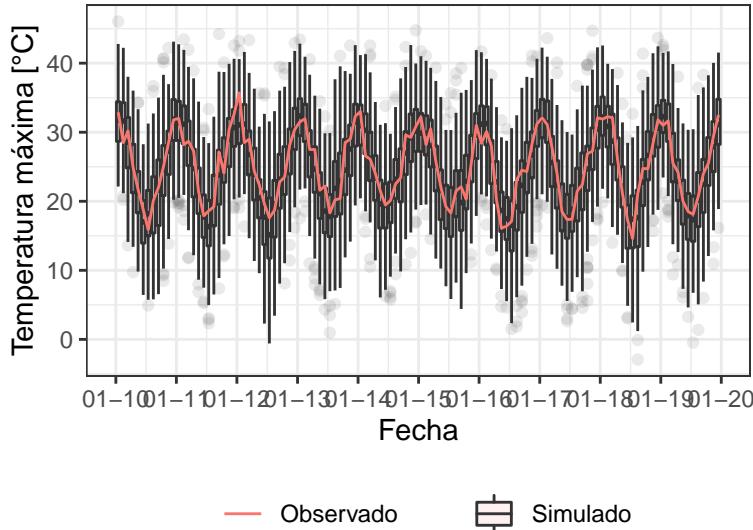


Figure 23: Comparación entre la temperatura máxima sintética y observada para el período 2010-2019 en la estación 87448.

891 5.3.4 Series sintéticas correlacionadas espacialmente

892 Una tercera alternativa para la generación de datos sobre estaciones meteorológicas com-
893 bina las anteriores mostradas pero generando el tiempo local con métodos que contemplan la
894 autocorrelación espacial. Para utilizar esta alternativa se deben disponer de más de una
895 sola estación porque de otro modo no se podrían calcular los parámetros de los variogramas
896 necesarios para la generación del tiempo local. Mientras más estaciones haya mejor, pero si
897 pueden generar campos espaciales confiables con alrededor de 10 puntos.

898 5.3.4.1 Crear archivos de entrada Para este ejemplo se utilizan datos de varias esta-
899 ciones pero el formato de los mismos es igual a los mostrados anteriormente.

900 Los archivos necesarios son:

- 901 • stations.csv
- 902 • climate.csv

903 Este ejemplo necesita de más estaciones ya que se debe ajustar un variograma. Por este

904 motivo, los datos usados en los dos ejemplos anteriores no son válidos.

905 A continuación se muestran las estaciones utilizadas en el ejemplo

```
# Vista de los metadatos de la estación  
knitr::kable(head(stations), "latex", booktabs = T) %>%  
  kableExtra::kable_styling(position = "center")
```

station_id	nombre	elev	geometry
86330	Artigas	120.38	POINT (547069 6636788)
86350	Rivera	241.94	POINT (639532.9 6580567)
86360	Salto	41.00	POINT (406863.6 6522326)
86430	Paysandú	61.12	POINT (402139.1 6420293)
86440	Melo	100.36	POINT (764393.7 6415079)
86460	Paso de los Toros	75.48	POINT (544004.4 6370787)

906 El objeto **stations** debe ser convertido de *tibble* a *sf*. El sistema de referencia espacial debe
907 ser planar. No es necesario un sistema de referencia espacial en particular, solamente las
908 coordenadas deben estar expresadas en metros.

909 En el mapa de la Figura 24 se muestra la distribución de las estaciones meteorológicas usadas
910 en el ejemplo.

```
# Convertir el objeto con las estaciones a coordenadas geográficas  
stations_geo <- stations %>%  
  sf::st_transform(crs = 4326) %>%  
  dplyr::mutate(lat = sf::st_coordinates(.)[,2],  
    lon = sf::st_coordinates(.)[,1])
```

```

# Creación del mapa con la distribución de las estaciones

leaflet::leaflet(data = stations_geo) %>%
  leaflet::addTiles() %>%
  leaflet::setView(lat = -33, lng = -56, zoom = 5) %>%
  leaflet::addCircleMarkers(lat = ~lat, lng = ~lon, radius = 3,
                            fillColor = "#377eb8",
                            color = "#377eb8",
                            stroke = FALSE, fillOpacity = 1, opacity = 1,
                            popup = ~sprintf("<b>%s (%d)</b><br>Lat.: %.3f<br>Lon.: %.3f",
                                             nombre, station_id, lat, lon, elev))

```



Figure 24: Distribución espacial de las estaciones utilizadas en Uruguay.

- 911 La información climática se aloja en el archivo `climate.csv`. Este archivo contiene los datos
- 912 de las estaciones meteorológicas que serán usadas en el ajuste del modelo. Las variables deben
- 913 ser las mismas a las mostradas en los demás ejemplos.
- 914 A continuación se muestran las primeras cinco filas del dataset y los tipos de datos de cada
- 915 una de las variables.

```

knitr::kable(head(climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")

```

date	station_id	tmax	tmin	prcp
1981-01-01	86330	34.6	21.7	0.0
1981-01-02	86330	33.8	21.2	19.5
1981-01-03	86330	28.3	19.4	0.0
1981-01-04	86330	30.3	17.4	0.0
1981-01-05	86330	33.4	17.4	21.0
1981-01-06	86330	32.8	20.4	8.0

916 En total se utilizan 15 estaciones meteorológicas, 10 provistas por INUMET (Instituyo
917 Uruguayo de Meteorología) y 5 por INIA (Instituto Nacional de Investigación Agropecuaria).

Códigos de identificación de las estaciones usadas

```
unique(climate$station_id)
```

918 ## [1] 86330 86350 86360 86430 86440 86460 86490 86560
919 ## [9] 86565 86580 90000001 90000002 90000003 90000004 90000005

920 **5.3.4.2 Ajuste de los modelos** El ajuste de los modelos es igual que para los dos
921 ejemplos antes mostrados. Se utiliza la función `control_fit` para la configuración del ajuste
922 y `local_calibrate` para realizar el ajuste.

Creación del objeto de control del ajuste

```
control_fit <- gamwgen::local_fit_control(  
  prcp_occurrence_threshold = 0.1,
```

```

# Umbral para la definición de días húmedos
avbl_cores = 6,
# Cantidad de núcleos disponibles
planar_crs_in_metric_coords = 32721)
# Sistema de referencia espacial (en metros)

```

923 Este ejemplo se realizará utilizando covriables trimestrales pero es válido para series esta-
 924 cionarias.

```

# Agregación de valores diarios
seasonal_covariates <- gamwgen::summarise_seasonal_climate(
  climate,
  # Registro histórico de variables meteorológicas
  umbral_faltantes = 0.2)
# Cantidad de datos faltantes tolerable

```

925 En la Figura 25 se muestra la variación de la precipitación acumulada estival para las 15
 926 estaciones utilizadas.

```

# Gráfico con la precipitación estival para las 15 localidades
ggplot2::ggplot(data = seasonal_covariates %>%
  dplyr::filter(season == 1),
  ggplot2::aes(x = year, y = seasonal_prcp)) +
  ggplot2::geom_line() +
  ggplot2::facet_wrap(~station_id, ncol = 3) +
  ggplot2::theme_bw() +
  ggplot2::labs(x = 'Años', y = 'Precipitación estival [mm]')

```

927 Se observa en la Figura 25 que si bien hay cierta coincidencia, no todos los picos y valles ocur-
 928 ren en el mismo momento. Utilizar un método que genere el tiempo local considerando estas

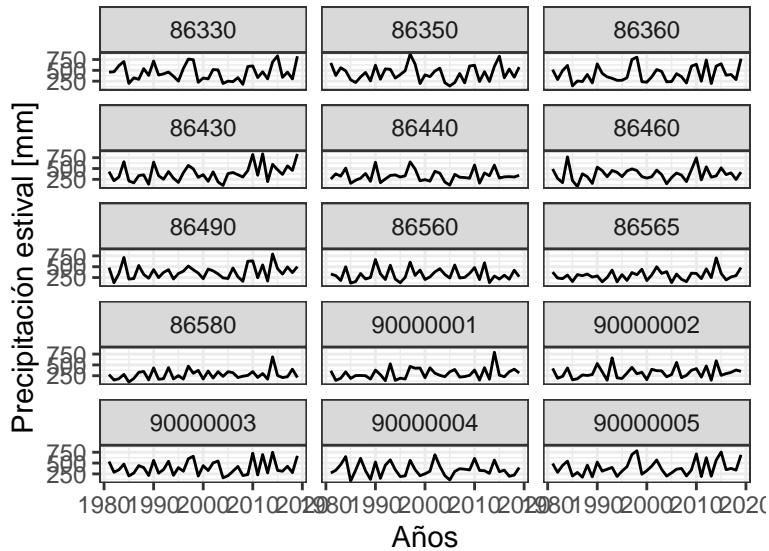


Figure 25: Precipitación acumulada estival para las 15 estaciones de Uruguay.

variaciones espaciales es muy útil para representar fehacientemente los patrones espaciales y temporales.

```
# Al correr la función se realiza el ajuste de los cuatro modelos
# para cada una de las estaciones. En este caso, por cuestiones de tiempo a cargar un
# Si el usuario desea correrlo deberá ver la nota anterior.

gamgen_fit <- gamwgen::local_calibrate(
  climate = climate,
  # Registro histórico de variables meteorológicas
  stations = stations,
  # Estaciones meteorológicas
  seasonal_covariates = seasonal_covariates,
  # Totales trimestrales de precipitación
  control = control_fit,
  # Objeto de control
  verbose = FALSE)

# Impresión de mensajes en la consola.
```

931 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```
# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/local/fit_local_correlated_weather.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_correlated_weather.RData",
                           package = "gamwgen"),
                new_path = 'input_data/local/fit_local_correlated_weather.Rdata')
}

# Se carga el archivo recientemente creado
load('input_data/local/fit_local_correlated_weather.RData')

# Clase del objeto con el ajuste del generador
class(gamgen_fit)

932 ## [1] "gamwgen"

# Contenido del modelo
names(gamgen_fit)

933 ## [1] "control"           "stations"          "climate"
934 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
935 ## [7] "fitted_models"       "models_data"       "models_residuals"
936 ## [10] "statistics_threshold" "exec_times"
```

937 Dentro del objeto se guardan todo lo necesario para la simulación así como información
938 accesoria.

- control: copia de la configuración usada para calibrar el generador

- **stations**: estaciones meteorológicas utilizadas para la calibración
- **climate**: datos climáticos de cada uno de las estaciones
- **seasonal_covariates**: series temporales de totales trimestrales de precipitación y medias trimestrales de temperaturas máxima y mínima.
- **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
- **start_climatology**: climatología diaria de cada una de las variables de entrada.
- **fitted_models**: modelos ajustados, uno para cada variable: temperaturas máxima y mínima y ocurrencia y montos de precipitación.
- **models_data**: datos usados efectivamente usados para ajustar los modelos (sin NAs)
- **models_residuals**: residuos de cada uno de los modelos. Es decir, la diferencia entre el valor ajustado por el modelo (clima local) y el valor observado en el día **i**
- **statistics_threshold**: umbrales de amplitud térmica diaria por mes. Si la amplitud simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de mantener la consistencia entre variables
- **exec_times**: tiempo de ejecución de cada una de las etapas del ajuste Al inspeccionar el objeto con los resultados se puede ver que cada estación meteorológica tiene su propia sublistas donde se almacenan los modelos para cada una de ellas. Cada uno de los modelos ajustados se pueden inspeccionar seleccionado por el código de identificación de cada estación meteorológica.

Modelos ajustados para cada una de las estaciones

```
names(gamgen_fit$fitted_models)
```

```
## [1] "86330"    "86350"    "86360"    "86430"    "86440"    "86460"
## [7] "86490"    "86560"    "86565"    "86580"    "90000001" "90000002"
## [13] "90000003" "90000004" "90000005"
```

```
summary(gamgen_fit$fitted_models$`86330`$tmax_fit)
```

```

962  ##
963  ## Family: gaussian
964  ## Link function: identity
965  ##
966  ## Formula:
967  ##  $tmax \sim s(tmax\_prev, tmin\_prev, k = 50) + s(prcp\_occ, bs = "re") +$ 
968  ##  $s(prcp\_occ\_prev, bs = "re") + s(doy, bs = "cc", k = 30) +$ 
969  ##  $s(SX1, SN1, k = 20) + s(SX2, SN2, k = 20) + s(SX3, SN3, k = 20) +$ 
970  ##  $s(SX4, SN4, k = 20)$ 
971  ##
972  ## Parametric coefficients:
973  ##          Estimate Std. Error t value Pr(>|t|)
974  ## (Intercept) 26.18758   0.03397   770.9 <2e-16 ***
975  ## ---
976  ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
977  ##
978  ## Approximate significance of smooth terms:
979  ##          edf Ref.df      F p-value
980  ##  $s(tmax\_prev,tmin\_prev)$  32.4286 40.704 235.57 <2e-16 ***
981  ##  $s(prcp\_occ)$            0.9979  1.000 520.81 <2e-16 ***
982  ##  $s(prcp\_occ\_prev)$      0.9987  1.000 866.16 <2e-16 ***
983  ##  $s(doy)$               10.3038 28.000  33.61 <2e-16 ***
984  ##  $s(SX1,SN1)$           3.6312  4.458  26.92 <2e-16 ***
985  ##  $s(SX2,SN2)$           2.0001  2.000  36.90 <2e-16 ***
986  ##  $s(SX3,SN3)$           4.3491  5.631  21.09 <2e-16 ***
987  ##  $s(SX4,SN4)$           2.0007  2.001  40.04 <2e-16 ***
988  ## ---

```

```

989 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
990 ##
991 ## R-sq.(adj) = 0.778 Deviance explained = 77.9%
992 ## fREML = 35496 Scale est. = 8.8219 n = 14128

993 A diferencia de los dos ejemplos anteriores, la lista fitted_models posee 15 elementos, uno
994 por cada estación meteorológica.

995 Con la creación del objeto gamgen_fit finaliza el proceso de ajuste y ya se pueden generar
996 series sintéticas para las estaciones usadas para calibrar el generador.

997 5.3.4.3 Generación de series La generación de series sigue la misma estructura ante-
998 rior, una función para configurar la generación y otra que realiza la generación propiamente
999 dicha. Cabe mencionar nuevamente que esta opción sólo es válida si en el ajuste y en la sim-
1000 ulación se usaron más de cinco estaciones meteorológicas diferentes. Con un menor número
1001 no es posible calcular los variogramas necesarios para la generación de los campos aleatorios.

```

```

# Creación del objeto de control
control_sim <- gamgen::local_simulation_control(
  nsim = 10,
  # Cantidad de simulaciones a realizar
  seed = 1234,
  # Semilla para que los resultados sean reproducibles
  avbl_cores = 6,
  # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = TRUE,
  # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = TRUE,
  # Guardar resultados intermedios para ahorrar RAM

```

```
remove_temp_files_used_to_save_ram = TRUE)  
# Borrar los resultados intermedios creados anteriormente
```

- 1002 Luego se procede a la simulación de datos meteorológicos. Los argumentos son los mismos
1003 que los mostrados en la sección anterior, sólo se modifica la función de control.

```
# Al correr la función se realiza la generación de series para  
# cada una de las estaciones.  
# En este caso, por cuestiones de tiempo, vamos a cargar un  
# objeto con los resultados de la simulación  
simulated_climate <- gamwgen::local_simulation(  
  model = gamgen_fit,  
  # Objeto con los resultados del ajuste  
  simulation_locations = stations,  
  # Estaciones para las cuales simular  
  start_date = as.Date('2019-01-01'),  
  # Fecha de comienzo de las simulaciones  
  end_date = as.Date('2019-12-31'),  
  # Fecha de fin de las simulaciones  
  control = control_sim,  
  # Objeto con la configuración  
  output_folder = getwd(),  
  # Directorio donde se guardarán los resultados  
  output_filename = 'simulations.csv',  
  # Nombre del archivo de salida  
  seasonal_covariates = seasonal_covariates,  
  # Covariables estacionales  
  verbose = FALSE)
```

```

# Impresión de mensajes en la consola

1004 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
1005 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

# Se copia el archivo preajustado a nuestro directorio de trabajo

if (!fs::file_exists('output_data/local/simulated_local_correlated_weather.RData')) {

  fs::file_copy(system.file('/autorun/local', "simulated_local_correlated_weather.RData",
                           new_path = 'output_data/local/simulated_local_correlated_weather.RData')
}

# Se carga el archivo recientemente creado

load('output_data/local/simulated_local_unconditional.RData')

# Clase del objeto con el ajuste del generador

class(simulated_climate)

1006 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo

names(simulated_climate)

1007 ## [1] "nsim"
1008 ## [2] "seed"
1009 ## [3] "realizations_seeds"
1010 ## [4] "simulation_points"
1011 ## [5] "output_file_with_results"
1012 ## [6] "output_file_fomart"
1013 ## [7] "rdata_file_with_fitted_stations_and_climate"
1014 ## [8] "exec_times"

```

1015 La lista contiene los siguientes objetos:

1016 • `nsim`: cantidad de realizaciones.

1017 • `seed`: semilla general para toda la generación. Corresponde a la que se incluye en la
1018 función de control.

1019 • `realization_seeds`: semillas para cada una de las realizaciones. Esto permite replicar
1020 los resultados.

1021 • `simulation_points`: puntos donde se generaron las series sintéticas.

1022 • `output_file_with_results`: nombre del archivo con los resultados.

1023 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.

1024 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
1025 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
1026 metadatos de cada uno de esos puntos.

1027 • `exec_times`: tiempo de ejecución de la generación.

1028 Como se observa, el objeto contiene los mismos elementos que los mostrados para otras
1029 configuraciones del “modelo local”.

1030 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones
1031 que forman parte de `local_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen
1032 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los
1033 parámetros del variograma. En este caso, al tratarse de tiempo local con dependencia es-
1034 pacial, los parámetros no son la media y escala de una distribución multivariada sino un
1035 variograma.

Calculo de los variogramas

```
gen_noise_params <- gamwgen:::generate_month_params(  
  residuals = gamgen_fit$models_residuals,  
  observed_climate = gamgen_fit$models_data,
```

```

stations = gamgen_fit$stations)

# Ejemplo de variograma de residuos de temperatura máxima para días secos

gen_noise_params[[1]]$variogram_parameters$tmax_dry

1036 ## [1] 0.000000 6.978337 495.779099

```

1037 Para cada una de las variables, precipitación y temperaturas máxima y mínima, se ajusta
 1038 un variograma por máxima verosimilitud que permite representar la variabilidad espacial de
 1039 los residuos de los modelos GAMs, que corresponden al tiempo local. Además, cada uno
 1040 de estos ajustes se realiza para cada mes del año para capturar la variabilidad estacional
 1041 de las variables. En el variograma se muestran los tres parámetros **nugget**, **psill** y **range**.
 1042 El rango de los datos es de 500 km, lo que es lógico si se considera que los puntos están
 1043 distribuidos homogeneamente en la República Oriental del Uruguay.

 1044 En la Figura ?? se muestra la variabilidad espacial del tiempo local de temperatura del día
 1045 1 de enero de 2019 para días secos. Cabe mencionar que se crean también los mismos datos
 1046 para los días húmedos y en función de la ocurrencia de precipitación se selecciona uno u otro.

```

# Configuración del paquete RandomFields

RandomFields::RFoptions(printlevel = 0, spConform = FALSE)

# Creación del campo de tiempo local espacialmente correlacionado

temp_local <- control_sim$temperature_noise_generating_function(
  simulation_points = stations %>%
    dplyr::mutate(latitude = sf::st_coordinates(.)[,2],
                 longitude = sf::st_coordinates(.)[,1]),
  gen_noise_params = gen_noise_params,
  month_number = 1L,

```

```

selector = 'Dry',
seed = NULL) %>%
dplyr::mutate(date = as.Date(paste0('2019-', '01', '-01'))) %>%
tidyr::gather(variable, valor, -c('date', 'geometry'))

# Descarga del mapa de Uruguay
uruguay <- raster::getData('GADM', country='URY',
                             level=1, download = FALSE) %>%
sf::st_as_sf(.) %>%
sf::st_transform(crs = 32721)

# Paleta de colores
colr <- grDevices::colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd'))

# Quiebres de la escala de colores
quiebres <- c(-8.0, -6.0, -4.0, -2.0, 0, 2.0, 4.0, 6.0, 8.0)

# Etiquetas para los colores
etiquetas <- c('-8', '-6', '-4', '-2', '0', '2', '4', '6', '8')

# Etiquetas de las facetas
labs <- c("Temp. Máx", "Temp. Min.")
names(labs) <- c("tmax_residuals", "tmin_residuals")

# Gráfico de tiempo local de temperatura máxima
ggplot2::ggplot(data = temp_local) +
  ggplot2::geom_sf(ggplot2::aes(color = valor), size = 4) +
  ggplot2::scale_color_gradientn(name = "Tiempo local [°C]",
                                 colours = colr(9),
                                 breaks = quiebres,

```

```

    labels = etiquetas,
    limits = c(-8, 8)) +
ggplot2::geom_sf(data = uruguay, fill = NA,
                  color = "black", inherit.aes = FALSE) +
ggplot2::facet_wrap(~variable,
                    labeller = ggplot2::labeller(variable = labs)) +
ggplot2::theme_bw()

```

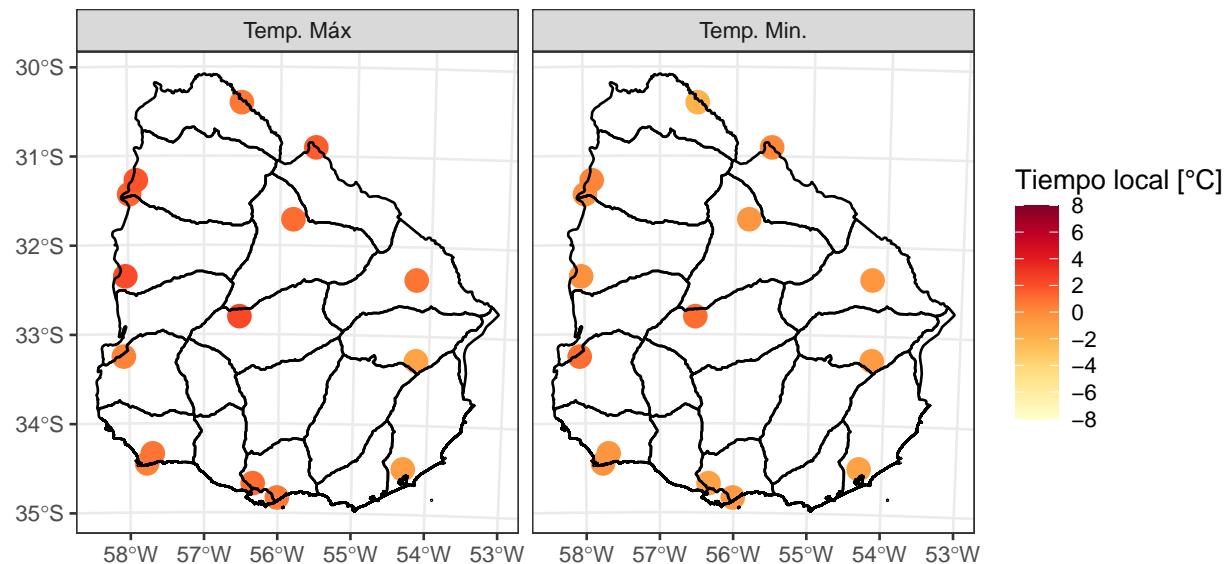


Figure 26: Tiempo local espacialmente correlacionado de temperatura máxima del 1 de enero de 2019 para las estaciones de Uruguay.

¹⁰⁴⁷ Se puede observar como el tiempo local tiene una estructura espacial que es representada a

1048 través del variograma y permite que los resultados para las distintas estaciones sean espaciamente consistentes. Esto quiere decir que estaciones meteorológicas cercanas tenderán a tener valores de tiempo local similares.

1051 5.4 Generación de series sintéticas para una grilla regular

1052 La generación de series sobre grillas regular es muy similar a lo mostrado anteriormente.
1053 Los fundamentos son los mismos, sólo los modelos estadísticos que modelan el clima local
1054 incorporan un nuevo término que permite incluir la dimensión espacial. para la utilización
1055 de esta variante del generador se deben utilizar la funciones de ajuste y generación que tienen
1056 el sufijo `spatial`, `spatial_calibrate` y `spatial_simulation`.

1057 **5.4.0.1 Crear archivos de entrada** Para este ejemplo se utilizan datos de varias estaciones pero el formato de los mismos es igual al mostrado anteriormente.

1059 Los archivos necesarios son:

- 1060 • `stations.csv`
- 1061 • `climate.csv`

1062 La información climática se almacena en el archivo `climate.csv`.

1063 **5.4.0.2 Ajuste de los modelos** Se utiliza la función `spatial_control_fit` para la
1064 configuración del ajuste y `spatial_calibrate` para realizar el ajuste.

```
# Creación del objeto de control
control_fit <- gamwgen::spatial_fit_control(
  prcp_occurrence_threshold = 0.1,
  # Umbral para la definición de días húmedos
```

```

avbl_cores = 6,
# Cantidad de núcleos disponibles
planar_crs_in_metric_coords = 32721)
# Sistema de referencia espacial (en metros)

```

1065 Este ejemplo se realizará utilizando covriables trimestrales pero es válido para series esta-
 1066 cionarias.

Agregación de valores diarios

```
seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes =
```

1067 Se observa que la función `spatial_calibrate` tiene los mismos argumentos que
 1068 `local_calibrate`. La única diferencia es que se requieren al menos 10 estaciones
 1069 para que los GAMs logren converger y obtener un resultado satisfactorio. Este número
 1070 mínimo es tentativo ya que depende de la variabilidad espacial de los datos. Es posible que
 1071 en regiones más heterogéneas o con relieve más complejo este número mínimo sea más alto.

Al correr la función se realiza el ajuste de los cuatro modelos para cada una de
 # las estaciones. En este caso, por cuestiones de tiempo a cargar un objeto ya precalc
 # Si el usuario desea correrlo deberá ver la nota anterior.

```
gamgen_fit <- gamwgen::spatial_calibrate(
  climate = climate,
  # Registro histórico de variables meteorológicas
  stations = stations,
  # Estaciones meteorológicas
  seasonal_covariates = seasonal_covariates,
  # Totales trimestrales de precipitación
  control = control_fit,
```

```

# Objeto de control
verbose = FALSE)

# Impresión de mensajes en la consola.

```

1072 Para esta demostración, se carga el objeto con el ajuste del modelo ya realizado.

```

# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/spatial/fit_spatial_conditional.RData')) {
  fs::file_copy(system.file('/autorun/spatial', "fit_spatial_conditional.RData", package = 'gam'))
  new_path = 'input_data/spatial/fit_spatial_conditional.Rdata')
}

# Se carga el archivo recientemente creado
load('input_data/spatial/fit_spatial_conditional.RData')

# Clase del objeto con el ajuste del generador
class(gamgen_fit)

1073 ## [1] "gamwgen"

# Contenido del modelo
names(gamgen_fit)

1074 ## [1] "control"           "stations"          "climate"
1075 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
1076 ## [7] "fitted_models"       "models_data"       "models_residuals"
1077 ## [10] "statistics_threshold" "exec_times"

```

1078 Dentro del objeto se guarda todo lo necesario para la simulación así como información acce-

1079 Soria.

1080 A diferencia de la configuración que ajusta distintos modelos para cada estación meteorológica,
1081 en este caso se ajusta un sólo modelo para toda la región de interés. Al visualizar lo que
1082 está almacenado en la sublista `fitted_models` se ve que solo hay cuatro GAMs, dos para
1083 precipitación y dos para las temperaturas máxima y mínima.

```
# GAMs ajustados

names(gamgen_fit$fitted_models)

1084 ## [1] "prcp_occ_fit" "prcp_amt_fit" "tmax_fit"      "tmin_fit"

1085 Cada uno de los GAMs ajustados se almacenan en el objeto gamgen_fit y pueden ser
1086 evaluados con la función summary().

summary(gamgen_fit$fitted_models$tmax_fit)

1087 ##
1088 ## Family: gaussian
1089 ## Link function: identity
1090 ##
1091 ## Formula:
1092 ## tmax ~ te(tmax_prev, tmin_prev, longitude, latitude, d = c(2,
1093 ##           2), k = length(unique_stations)) + te(type_day, longitude,
1094 ##           latitude, d = c(1, 2), bs = c("re", "tp"), k = length(unique_stations)) +
1095 ##           te(type_day_prev, longitude, latitude, d = c(1, 2), bs = c("re",
1096 ##               "tp"), k = length(unique_stations)) + te(doy, longitude,
1097 ##               latitude, d = c(1, 2), bs = c("cc", "tp"), k = length(unique_stations)) +
1098 ##               te(SX1, SN1, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
1099 ##               te(SX2, SN2, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
1100 ##               te(SX3, SN3, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
```

```

1101 ##      te(SX4, SN4, longitude, latitude, d = c(2, 2), k = length(unique_stations))
1102 ##
1103 ## Parametric coefficients:
1104 ##              Estimate Std. Error t value Pr(>|t|)
1105 ## (Intercept) 25.91      29.52   0.878   0.38
1106 ##
1107 ## Approximate significance of smooth terms:
1108 ##                                         edf Ref.df      F p-value
1109 ## te(tmax_prev,tmin_prev,longitude,latitude) 76.902 86.484 1267.40 <2e-16 ***
1110 ## te(type_day,longitude,latitude)            27.680 29.000 161.22 <2e-16 ***
1111 ## te(type_day_prev,longitude,latitude)       14.703 15.000 726.24 <2e-16 ***
1112 ## te(doy,longitude,latitude)                47.907 195.000 77.44 <2e-16 ***
1113 ## te(SX1,SN1,longitude,latitude)           18.355 23.456 75.14 <2e-16 ***
1114 ## te(SX2,SN2,longitude,latitude)           6.005  6.009 184.32 <2e-16 ***
1115 ## te(SX3,SN3,longitude,latitude)           30.642 36.395 48.54 <2e-16 ***
1116 ## te(SX4,SN4,longitude,latitude)           6.002  6.004 193.99 <2e-16 ***
1117 ## ---
1118 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
1119 ##
1120 ## R-sq.(adj) = 0.779  Deviance explained = 77.9%
1121 ## fREML = 5.221e+05  Scale est. = 8.9846    n = 207269

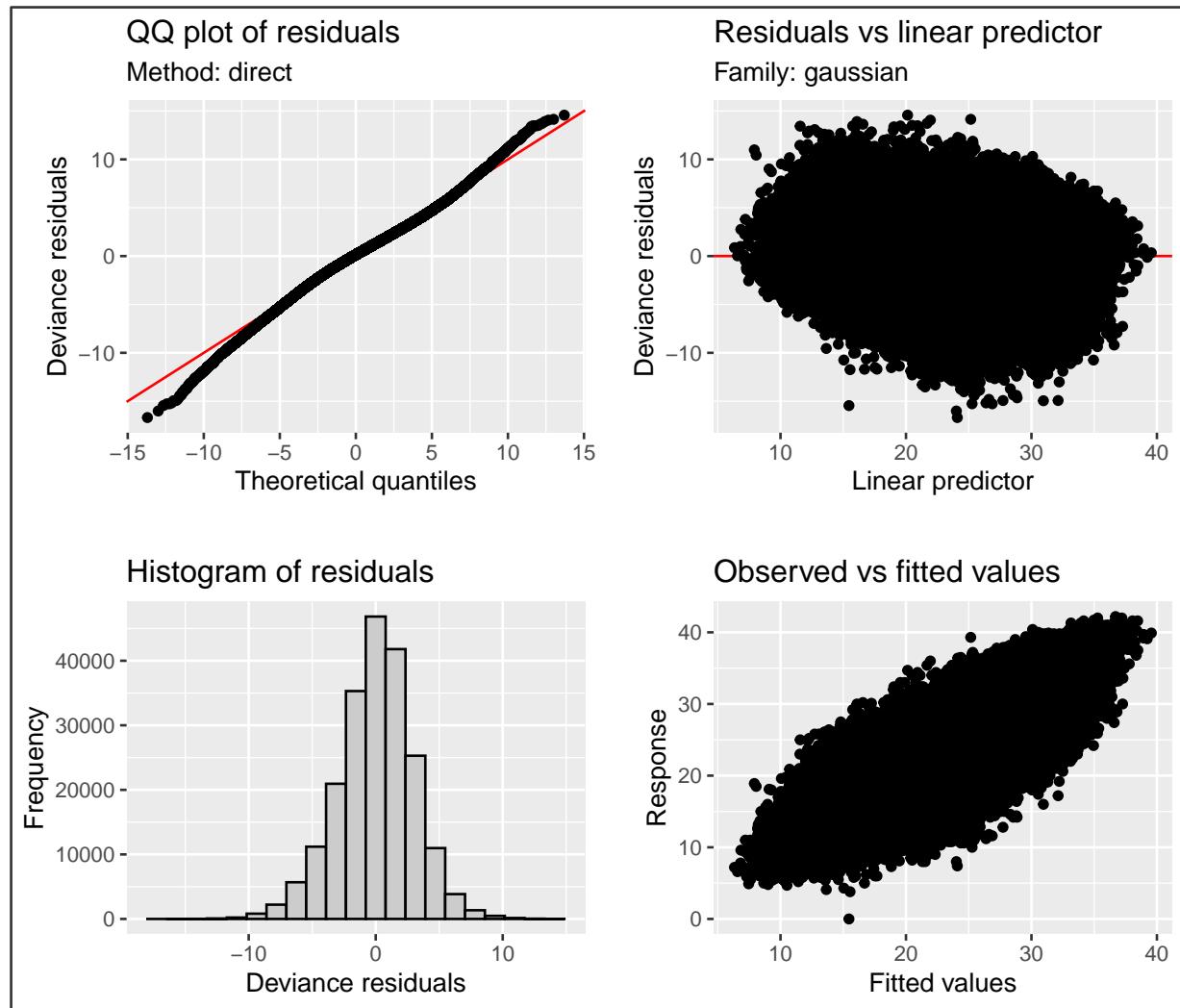
```

1122 La función `summary` permite analizar los resultados del ajuste de cada uno de los modelos.
 1123 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la
 1124 parte superior bajo el apartado `Formula` y la significancia de cada uno de los términos del
 1125 modelo en la tabla inmediatamente inferior. A diferencia del “*modelo local*”, las funciones
 1126 suavizadas no son splines sino productos tensores. Los productos tensores permiten combi-
 1127 nar más de una variable en la *función suavizada*. Por ejemplo, para el primer término de la

1128 fórmula `te(tmax_prev,tmin_prev,longitude,latitude)`, al incluir la latitud y longitud,
1129 la interacción entre las temperaturas máxima y mínima del día previo se modela espacial-
1130 mente. Cada valor de la matriz de datos de entrada se transforma en una superficie y se
1131 ajusta un función suavizada que varía espacialmente. Lo mismo sucede para cada uno de
1132 los términos del modelo. Los modelos espaciales también pueden ser diagnosticados con la
1133 función `appraise` aunque algunos diagnósticos serán difíciles de interpretar por la cantidad
1134 de valores que son usados para el ajuste de los modelos.

Diagnosticos gráficos

```
gratia::appraise(gamgen_fit$fitted_models$tmax_fit) +  
  ggplot2::theme_bw()
```



1135

1136 Estos diagnósticos exploratorios pueden aplicarse a cada uno de los modelos ajustados por la
1137 función `spatial_calibrate`.

1138 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
1139 series sintéticas para la o las estaciones usadas para calibrar el generador.

1140 **5.4.0.3 Generación de series** La generación de series sigue la misma estructura ante-
1141 rior, una función para configurar la generación y otra que realiza la generación propiamente
1142 dicha.

1143 La función de control debe configurarse de la siguiente manera:

```

control_sim <- gamwgen::spatial_simulation_control(
  nsim = 10, # Cantidad de simulaciones a realizar
  seed = 1234, # Semilla para que los resultados sean reproducibles
  avbl_cores = 6, # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = TRUE, # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = TRUE, # Guardar resultados intermedios para ahorrar espacio
  remove_temp_files_used_to_save_ram = TRUE) # Borrar los resultados intermedios creados

```

- 1144 Para simular sobre una grilla regular el argumento `use_spatially_correlated_noise` debe
 1145 ser TRUE, de otra manera, se produciría un error y la generación se abortaría.

Agregación de valores diarios

```
seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes =
```

- 1146 Para simular sobre una grilla regular se debe crear un objeto de tipo `sf` con puntos
 1147 equidistantes entre sí. El paquete `sf` cuenta con una función muy útil para esta tarea,
 1148 `st_make_grid`. En el presente ejemplo se genera una grilla de 25 km de lado a partir de un
 1149 shapefile con los límites administrativos de la República Oriental del Uruguay. La Figura
 1150 ?? muestra la distribución de los puntos generados. Cabe mencionar que la grilla no debe
 1151 ser de tipo polígono sino puntos.

Shapefile de Uruguay

```
uruguay <- raster::getData('GADM', country='URY',
  level = 1, download = FALSE) %>%
  sf::st_as_sf(.) %>%
  sf::st_transform(crs = 32721)
```

Creación de la grilla sobre la que se simulará

```

grilla_simulacion_centers <- uruguay %>%
  dplyr::select(geometry) %>%
  sf::st_transform(gamgen_fit$crs_used_to_fit) %>%
  sf::st_make_grid(cellsize = 25000, what = 'centers') %>%
  sf::st_sf(grid_id = 1:length(.), geometry = .) %>%
  sf::st_intersection(uruguay)

ggplot2::ggplot() +
  ggplot2::geom_sf(data = grilla_simulacion_centers) +
  ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw()

```

1152 No hay límites para la resolución espacial de la grilla pero el usuario debe ser consciente que
 1153 cada vez que un píxel se divide a la mitad se generan cuatro puntos. Esto implica que la
 1154 memoria disponible puede rápidamente agotarse y el proceso terminar abruptamente.

1155 Al utilizar covariables trimestrales es necesario que éstas se interpolen para cada uno de
 1156 los puntos de la grilla. A continuación se muestran dos ejemplos, uno para precipitación
 1157 acumulada para el trimestre estival de 2019 y la temperatura máxima media del mismo
 1158 período. La Figura 28 muestra el resultado de la interpolación de la precipitación observada
 1159 del trimestre estival de 2019.

```

# Interpolar covariables sobre la grilla de simulación

seasonal_covariates <-
  gamwgen:::get_covariates(model = gamgen_fit,
                           simulation_points = grilla_simulacion_centers,
                           seasonal_covariates,
                           simulation_dates,
                           control = control_sim)

```

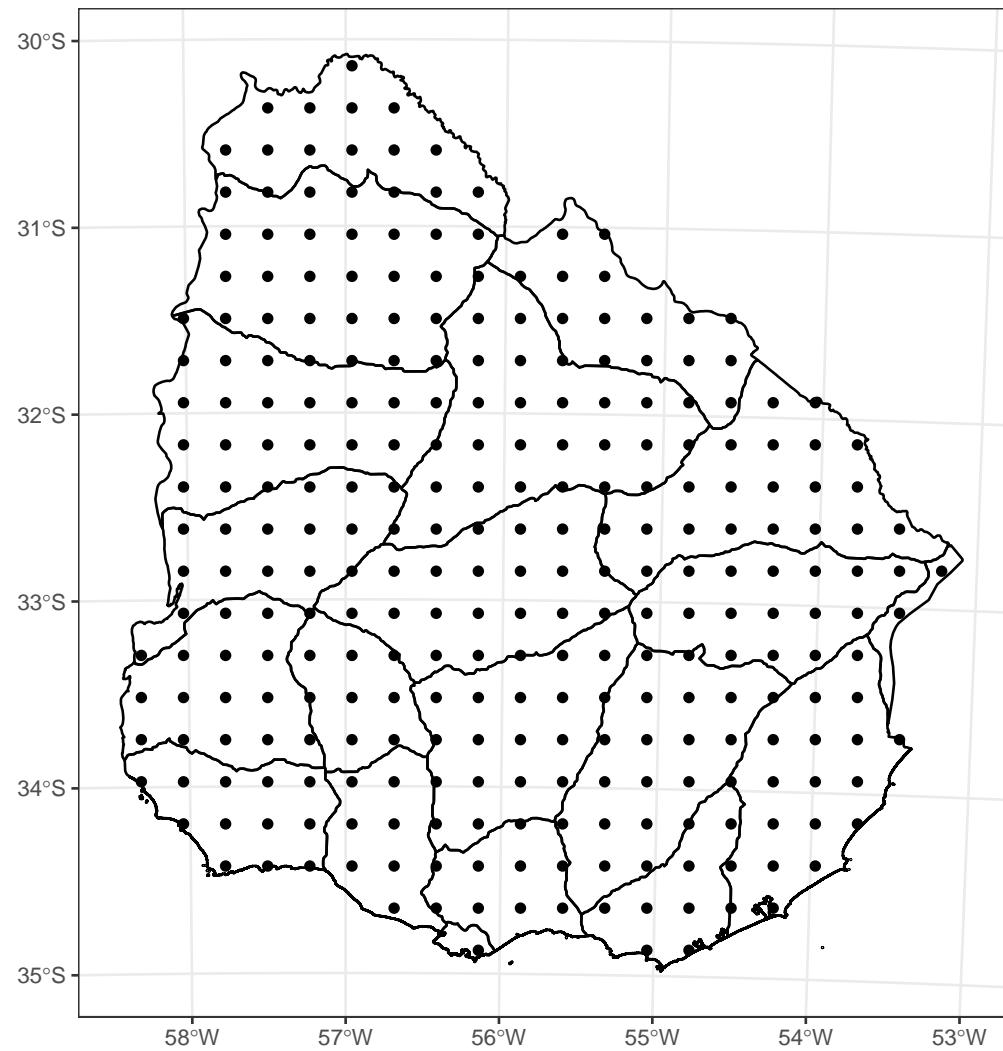


Figure 27: Grilla regular sobre Uruguay.

```

# Crear data.frame con los resultados

seasonal_prcp <- gamwgen:::sf2raster(seasonal_covariates %>%
                                         dplyr::filter(year == 2019, season == 1),
                                         variable = 'seasonal_prcp') %>%
                                         raster::as.data.frame(., xy = TRUE) %>%
                                         dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
                                         dplyr::mutate(., x = longitude, y = latitude) %>%
                                         sf::st_as_sf(., coords = c('x', 'y')) %>%
                                         sf::st_set_crs(., 32721) %>%
                                         sf::st_intersection(uruguay) %>%
                                         dplyr::select(latitude, longitude, z, geometry)

# Definición de colores de la paleta

colores <- colorRampPalette(c("#cccccc", "#7bccc4", "#2b8cbe", "#0868ac",
                               "#084081", "#807dba", "#6a51a3", "#54278f",
                               "#3f007d", "#e7298a", "#ce1256", "#67001f"))

# Definicion de quiebres de la paleta

quiebres <- c(0.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0,
              70.0, 80.0, 90.0, 100.0, 110.0, 120.0, 130.0,
              140.0, 150.0, 160.0)

# Definición de etiquetas

etiquetas <- c('0', '5', '10', '20', '30', '40', '50', '60',
               '70', '80', '90', '100', '110', '120', '130',
               '140', '150', '160')

ggplot(data = seasonal_prcp,
        ggplot2::aes(x = longitude, y = latitude, fill = z)) +
        geom_tile(colour="black") +

```

```

scale_fill_gradientn(name = "Precipitation [mm]", colours = colores(17),
                     breaks = quiebres, labels = etiquetas) +
ggplot2::geom_sf(data = uruguay, fill = NA,
                  color = "black", inherit.aes = FALSE) +
ggplot2::theme_bw() +
ggplot2::theme(panel.background = element_rect(fill = "ivory"),
               axis.text = element_text(size = 11, colour = 1),
               legend.key.height = unit(2.5, "cm"))

```

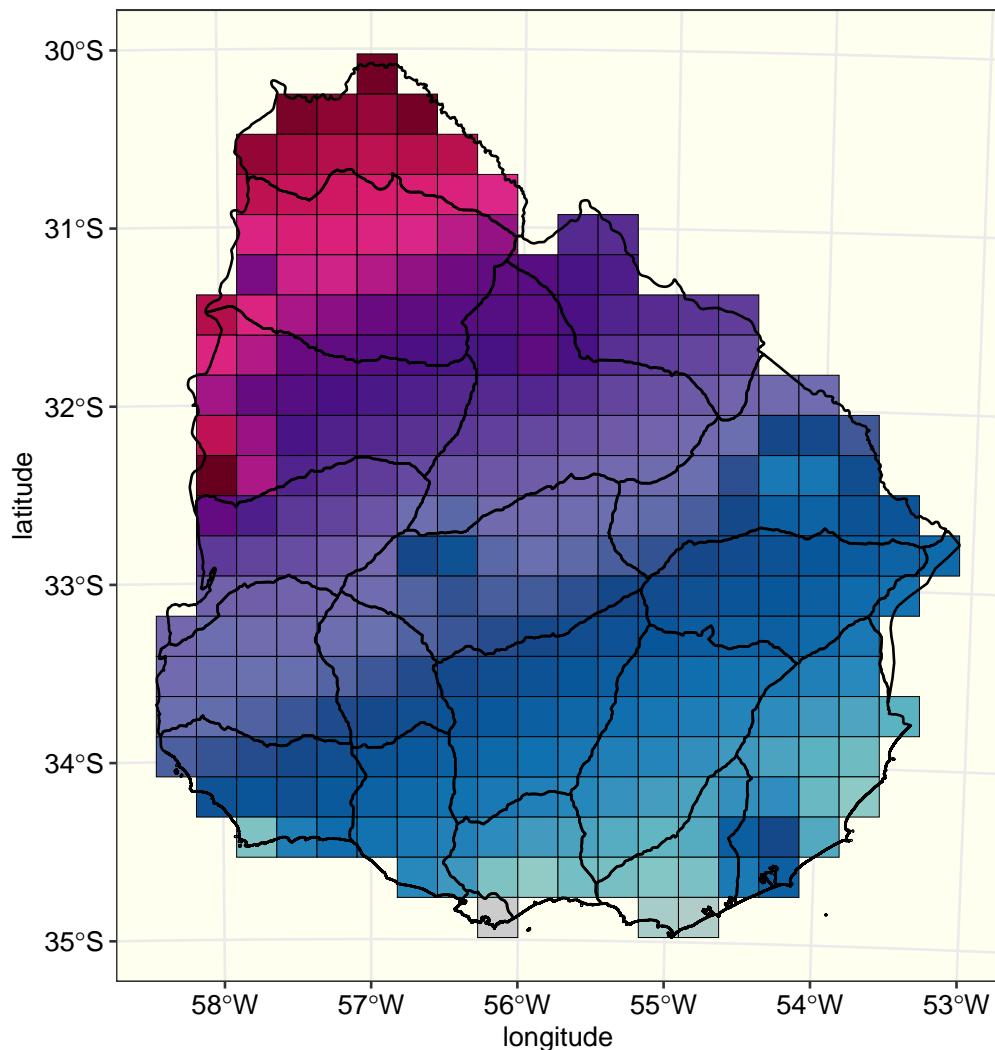


Figure 28: Precipitación acumulada del verano de 2019 sobre Uruguay.

- 1160 La Figura 29 muestra la temperatura máxima media para el mismo período.

```
seasonal_tmax <- gamwgen:::sf2raster(seasonal_covariates %>%
                                         dplyr::filter(year == 2019, season == 1),
                                         variable = 'seasonal_tmax') %>%
                                         raster::as.data.frame(., xy = TRUE) %>%
                                         dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
                                         dplyr::mutate(., x = longitude, y = latitude) %>%
                                         sf::st_as_sf(., coords = c('x', 'y')) %>%
                                         sf::st_set_crs(., 32721) %>%
                                         sf::st_intersection(uruguay) %>%
                                         dplyr::select(latitude, longitude, z, geometry)

# Definición de colores de la paleta
colr <- colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd')) 

# Definición de quiebres de la paleta
quiebres <- c(0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0)

# Definición de etiquetas
etiquetas <- c('0', '5', '10', '15', '20', '25', '30', '35', '40', '45')

ggplot(data = seasonal_tmax,
        ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  geom_tile(colour="black") +
  scale_fill_gradientn(name = "Temperatura máxima [°C]",
                        colours = colr(9),
                        breaks = quiebres,
                        labels = etiquetas) +
  ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw()
```

```

ggplot2:: labs(title = "", x = "", y = "") +
ggplot2::theme(panel.background = element_rect(fill = "ivory"),
               axis.text = element_text(size = 11, colour = 1))

```

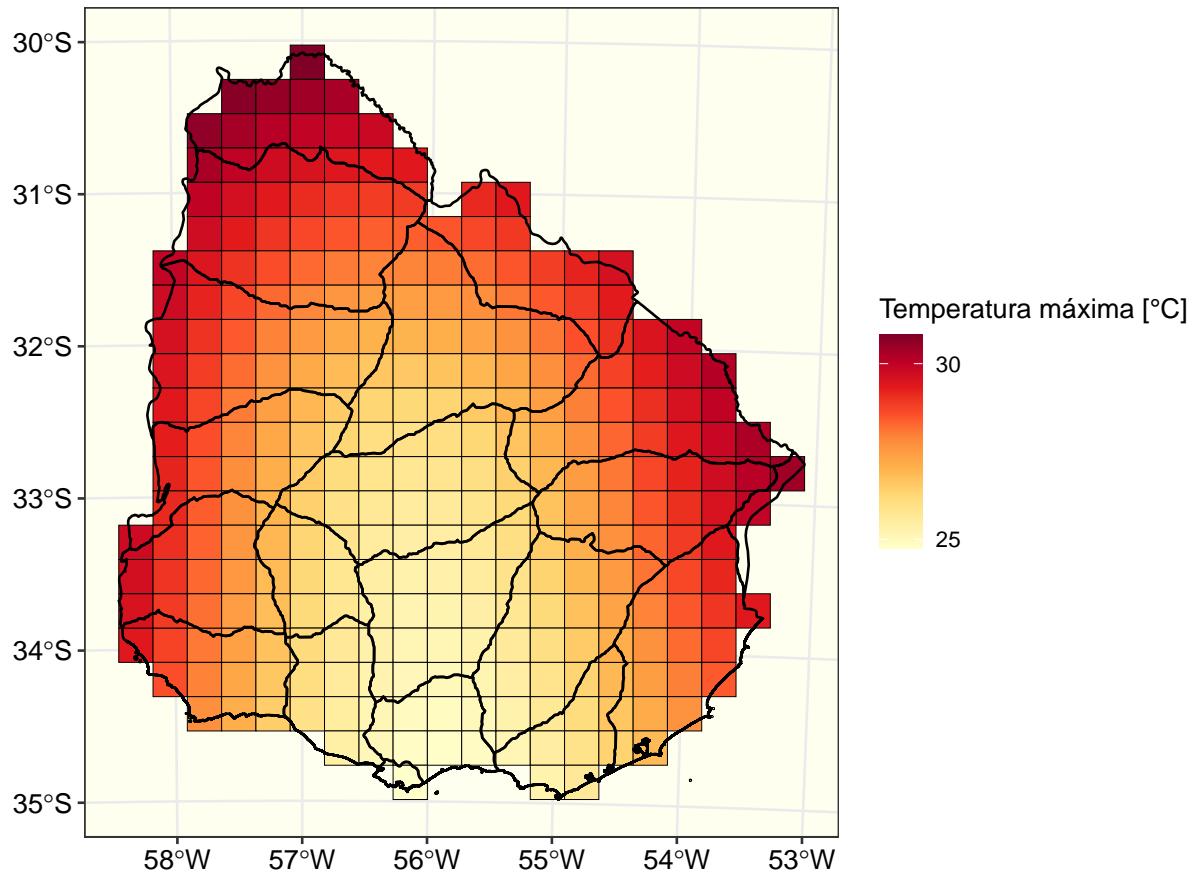


Figure 29: Temperatura máxima media del verano de 2019 sobre Uruguay.

1161 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de

1162 simulación son:

1163 • **model**: objeto con el resultado de la función `local_calibrate()`

1164 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simular. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es

1166 necesario que todas estén presentes, se pueden generar series solo sobre algunas de
 1167 ellas.
 1168 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron
 1169 covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria.
 1170 Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de
 1171 covariables, ni tampoco posterior. Se debe introducir una fecha en formato **date**
 1172 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron co-
 1173 variaciones estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso
 1174 contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables,
 1175 tampoco puede ser posterior. Se debe introducir una fecha en formato **date**
 1176 • **control**: objeto de control creado con la función **control_sim()**.
 1177 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales
 1178 como intermedios.
 1179 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad,
 1180 el archivo generado es un archivo de texto en formato separado por comas (.csv)
 1181 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con co-
 1182 variaciones, la generación también debe realizarse con ellas. Caso contrario se producirá
 1183 un error. Se debe introducir un data frame con los valores agregados para las tres
 1184 variables (precipitación y temperaturas máxima y mínima) pero no necesariamente
 1185 deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de
 1186 algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar
 1187 estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma
 1188 longitud que el período a generar
 1189 • **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```

# Al correr la función se realiza la generación de series para cada una de las estacio-
# En este caso, por cuestiones de tiempo, vamos a cargar un objeto con los resultados
simulated_climate <- gamwgen::spatial_simulation(

```

```

model = gamgen_fit,
# Objeto con los resultados del ajuste
simulation_locations = grilla_simulacion_centers,
# Estaciones para las cuales simular
start_date = as.Date('2019-01-01'),
# Fecha de comienzo de las simulaciones
end_date = as.Date('2019-01-31'),
# Fecha de fin de las simulaciones
control = control_sim,
# Objeto con la configuración
output_folder = getwd(),
# Directorio donde se guardarán los resultados
output_filename = 'simulations.csv',
# Nombre del archivo de salida
seasonal_covariates = seasonal_covariates,
# Covariables estacionales
verbose = FALSE)
# Impresión de mensajes en la consola

```

1190 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
 1191 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```

# Se copia el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('output_data/spatial/simulated_spatial_conditional.RData')) {
  fs::file_copy(system.file('/autorun/spatial', "simulated_spatial_conditional.RData",
                           new_path = 'output_data/spatial/simulated_spatial_conditional.RData')
}
# Se carga el archivo recientemente creado

```

```

load('output_data/spatial/simulated_spatial_conditional.RData')

# Clase del objeto con el ajuste del generador

class(simulated_climate)

1192 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo

names(simulated_climate)

1193 ## [1] "nsim"
1194 ## [2] "seed"
1195 ## [3] "realizations_seeds"
1196 ## [4] "simulation_points"
1197 ## [5] "output_file_with_results"
1198 ## [6] "output_file_fomart"
1199 ## [7] "rdata_file_with_fitted_stations_and_climate"
1200 ## [8] "exec_times"

```

1201 La lista contiene los siguientes objetos:

- 1202 • **nsim**: cantidad de realizaciones.
- 1203 • **seed**: semilla general para toda la generación. Corresponde a la que se incluye en la función de control.
- 1204 • **realization_seeds**: semillas para cada una de las realizaciones. Esto permite replicar los resultados.
- 1205 • **simulation_points**: puntos donde se generaron las series sintéticas.
- 1206 • **output_file_with_results**: nombre del archivo con los resultados.
- 1207 • **output_file_format**: tipo de archivo de salida, en este caso .csv.

- 1210 • `rdata_file_with_fitted_stations_and_climate`: archivo .RData con los datos me-
 1211 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
 1212 metadatos de cada uno de esos puntos.
- 1213 • `exec_times`: tiempo de ejecución del la generación.

1214 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

1215 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones
 1216 que forman parte de `spatial_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen
 1217 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los
 1218 variogramas que dan origen a los campos gaussianos.

1219 En la tabla anterior se muestran los parámetros necesarios para generar el tiempo local de
 1220 las temperaturas máxima y mínima.

```
gen_noise_params <- gamwgen:::generate_month_params(  

  residuals = gamgen_fit$models_residuals,  

  observed_climate = gamgen_fit$models_data,  

  stations = gamgen_fit$stations)
```

Ejemplo de variograma de residuos de temperatura máxima para días secos

```
gen_noise_params[[1]]$variogram_parameters$tmax_dry
```

1221 ## [1] 0.000000 7.045242 493.298928

1222 Explicar variograma.

1223 Los parámetros para cada uno de los meses permiten generar valores de tiempo local para
 1224 las dos temperaturas a partir de una distribución normal multivariada.

1225 A continuación se muestra un ejemplo para el año 2019 sólo para los días lluviosos.

```

temp_local    <- control_sim$temperature_noise_generating_function(
  simulation_points = grilla_simulacion_centers,
  gen_noise_params = gen_noise_params,
  month_number = 1L,
  selector = 'Dry',
  seed = 1234) %>%
  dplyr::mutate(date = as.Date(paste0('2019-', '01', '-', '01'))) %>%
  dplyr::filter(date == as.Date('2019-01-01')) %>%
  tidyverse::gather(variable, valor, -c('geometry'))

tmax_residuals <- gamwgen:::sf2raster(temp_local %>%
  dplyr::filter(variable == 'tmax_residuals'),
  'valor') %>%
  raster::as.data.frame(., xy = TRUE) %>%
  dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
  dplyr::mutate(., x = longitude, y = latitude) %>%
  sf::st_as_sf(., coords = c('x', 'y')) %>%
  sf::st_set_crs(., 32721) %>%
  sf::st_intersection(uruguay) %>%
  dplyr::select(latitude, longitude, z, geometry)

colr <- colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd'))

quiebres <- c(-4.0, -3, -2, -1, 0, 1, 2, 3, 4)

etiquetas <- c('-4', '-3', '-2', '-1', '0', '1', '2', '3', '4')

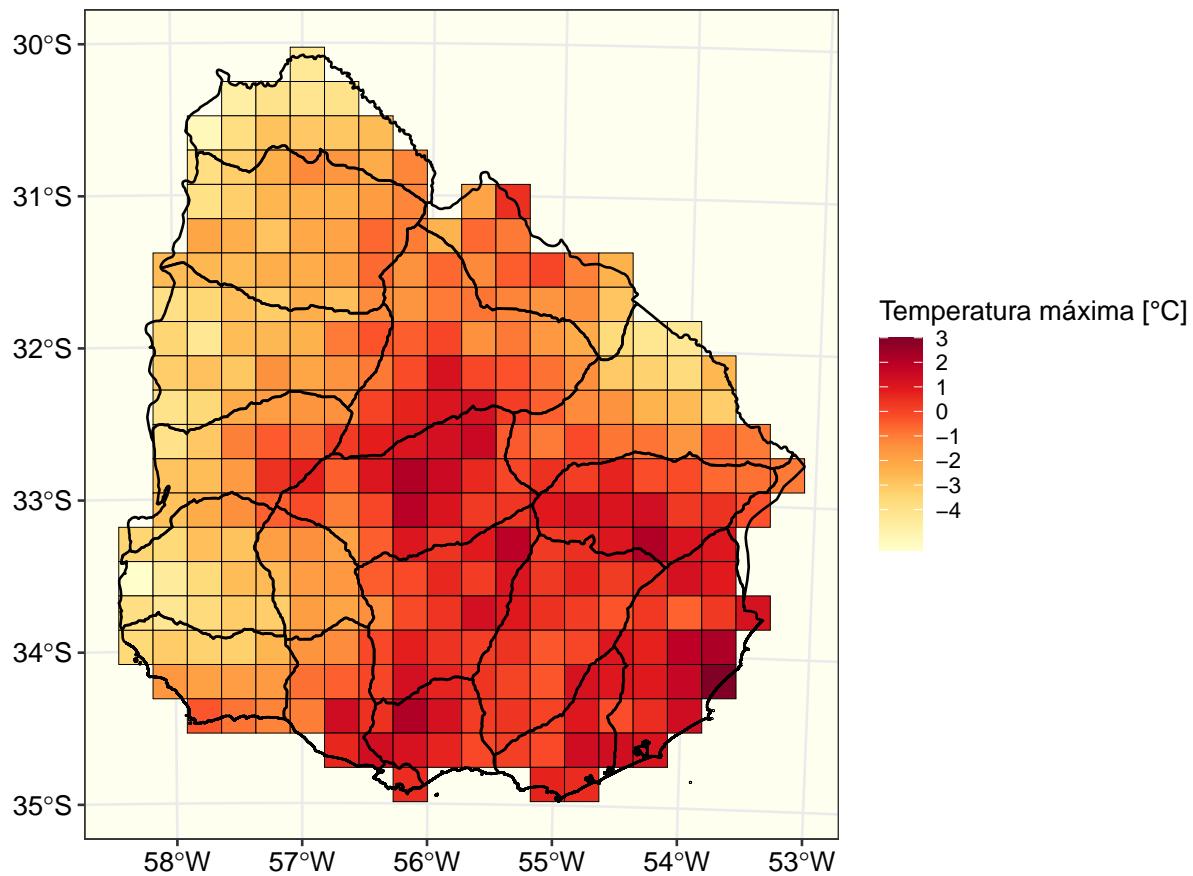
```

```

ggplot(data = tmax_residuals,
        ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  geom_tile(colour="black") +
  scale_fill_gradientn(name = "Temperatura máxima [°C]",
                        colours = colr(9),
                        breaks = quiebres,
                        labels = etiquetas) +
  ggplot2::geom_sf(data = uruguay, fill = NA,
                    color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw() +
  ggplot2:: labs(title = "", x = "", y = "") +
  ggplot2::theme(panel.background = element_rect(fill = "ivory"),
                 axis.text = element_text(size = 11, colour = 1))

```

1226



1227

1228 El mapa muestra el campo Gaussiano de temperatura máxima de los días secos para el
 1229 primer día de la simulación, 1 de enero de 2019.

Se carga el set de datos simulados

```
simulated_climate <- readr::read_csv(here::here('output_data/spatial/simulated_spatial_0
```

Primeras filas del objeto de salidas

```
knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
```

```
kableExtra::kable_styling(position = "center",
                           latex_options = c("striped", "scale_down"))
```

realization	point_id	longitude	latitude	date	tmax	tmin	prcp
1	1	453759.4	6590418	2019-01-01	33.03589	19.72844	1.826571
1	2	553759.4	6590418	2019-01-01	31.77257	19.65962	27.883864
1	3	578759.4	6590418	2019-01-01	32.71455	20.47479	32.879767
1	4	428759.4	6615418	2019-01-01	30.64363	16.02136	7.032041
1	5	453759.4	6615418	2019-01-01	31.83107	18.12258	14.671730
1	6	478759.4	6615418	2019-01-01	33.32291	18.75998	14.781878

1230 El resultado de la generación es un archivo .csv que contiene la siguiente información:

- 1231 • **realization**: número de realización. Es un valor entero entre 1 y la cantidad de
- 1232 realizaciones definida por el usuario.
- 1233 • **station_id**: número único de identificación de la estación meteorológica o del punto
- 1234 arbitrario.
- 1235 • **date**: fechas de cada uno de los días de la simulación.
- 1236 • **tmax**: valores de temperatura máxima generada expresada en °C.
- 1237 • **tmin**: valores de temperatura mínima generada expresada en °C.
- 1238 • **prcp**: valores de precipitación diaria generada expresada en mm.

1239 La siguiente Figura muestra un ejemplo de las series de precipitación diaria para el 1 de

1240 enero de 2019 para la primera realización.

```
simulated_climate_geo <- simulated_climate %>%
  dplyr::filter(date == as.Date('2019-01-01'),
                realization == 1) %>%
  sf::st_as_sf(coords = c('longitude', 'latitude'), crs = 32721) %>%
  gamwgen::sf2raster(., 'prcp') %>%
```

```

raster::as.data.frame(., xy = TRUE) %>%
dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
dplyr::mutate(., x = longitude, y = latitude) %>%
sf::st_as_sf(., coords = c('x', 'y')) %>%
sf::st_set_crs(., 32721) %>%
sf::st_intersection(uruguay) %>%
dplyr::select(latitude, longitude, z, geometry)

1241 ## Warning: attribute variables are assumed to be spatially constant throughout all
1242 ## geometries

colores <- colorRampPalette(c("#cccccc", "#7bccc4", "#2b8cbe", "#0868ac",
                           "#084081", "#807dba", "#6a51a3", "#54278f",
                           "#3f007d", "#e7298a", "#ce1256", "#67001f"))

quiebres <- c(0.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0,
             60.0, 70.0, 80.0, 90.0, 100.0, 110.0,
             120.0, 130.0, 140.0, 150.0, 160.0)

etiquetas <- c('0', '5', '10', '20', '30', '40', '50',
               '60', '70', '80', '90', '100', '110',
               '120', '130', '140', '150', '160')

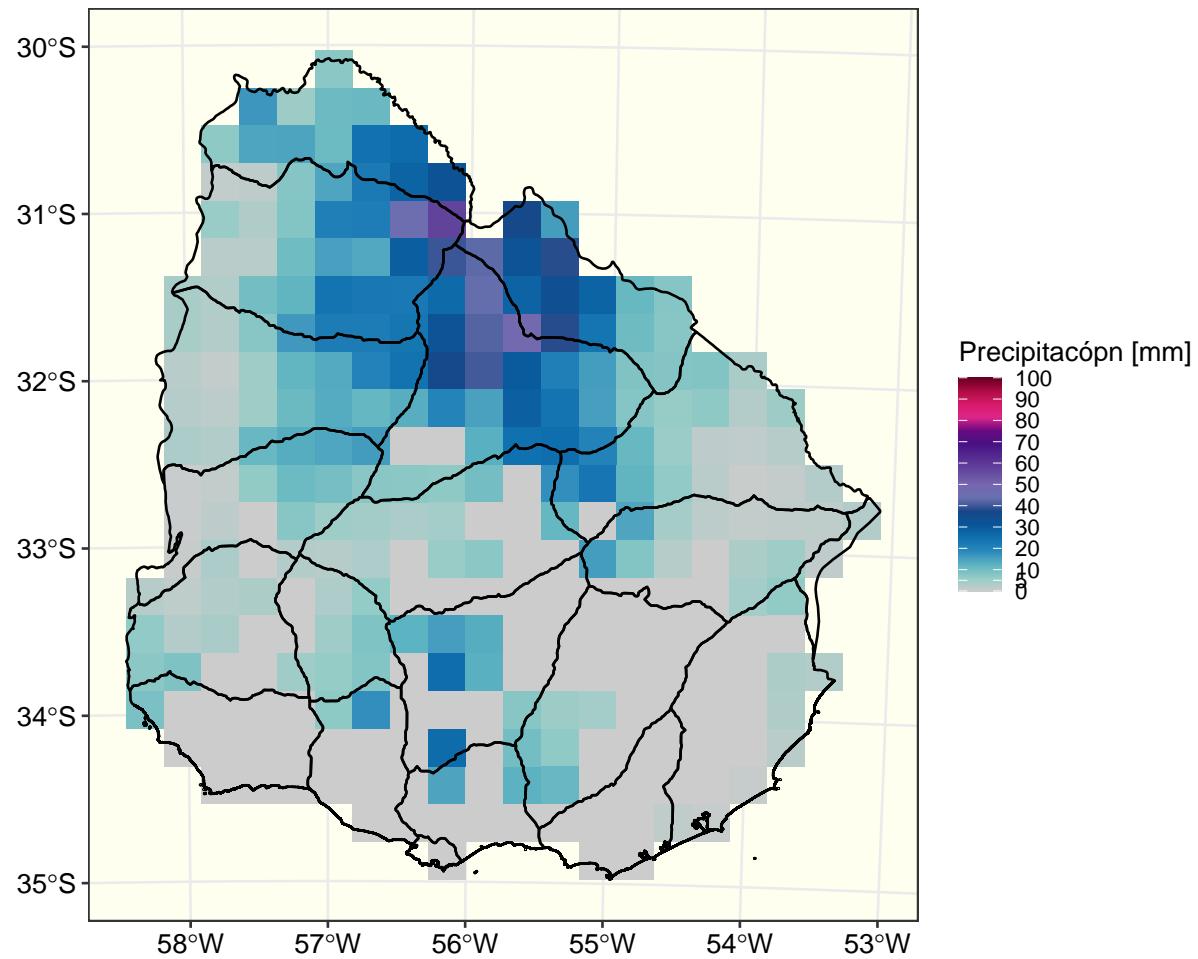
ggplot2::ggplot(data = simulated_climate_geo) +
  ggplot2::geom_tile(ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  ggplot2::scale_fill_gradientn(name = "Precipitacón [mm]",
                                colours = colores(17),
                                breaks = quiebres,

```

```

        labels = etiquetas,
        limits = c(0, 100)) +
ggplot2::geom_sf(data = uruguay, fill = NA,
                  color = "black", inherit.aes = FALSE) +
ggplot2::theme_bw() +
ggplot2:: labs(title = "", x = "", y = "") +
ggplot2::theme(panel.background = element_rect(fill = "ivory"),
               axis.text = element_text(size = 11, colour = 1))

```



₁₂₄₄ **References**

- ₁₂₄₅ Kleiber, W., Katz, R.W., Rajagopalan, B., 2012. Daily spatiotemporal precipitation simulation using latent and transformed gaussian processes. *Water Resources Research* 48.
- ₁₂₄₆
- ₁₂₄₇ Kleiber, W., Katz, R.W., Rajagopalan, B., 2013. Daily minimum and maximum temperature simulation over complex terrain. *Ann. Appl. Stat.* 7, 588–612.
- ₁₂₄₈
- ₁₂₄₉ Schlather, M., Malinowski, A., Menck, P.J., Oesting, M., Strokorb, K., 2015. Analysis, simulation and prediction of multivariate random fields with package randomfields. *Journal of Statistical Software* 63.
- ₁₂₅₀
- ₁₂₅₁
- ₁₂₅₂ Simpson, G., 2018. Introducing *gratia*. From the Bottom of the Heap.
- ₁₂₅₃ Verdin, A., 2016. Stochastic space-time modeling for agricultural decision support in the argentine pampas (Thesis).
- ₁₂₅₄
- ₁₂₅₅ Wickham, H., 2011. Ggplot2. *Wiley Interdisciplinary Reviews: Computational Statistics* 3,
- ₁₂₅₆ 180–185.
- ₁₂₅₇ Wood, S., 2015. Package “mgcv”. R package version 1, 29.
- ₁₂₅₈ Wood, S.N., 2017. Generalized additive models: An introduction with r. Chapman;
- ₁₂₅₉ Hall/CRC.